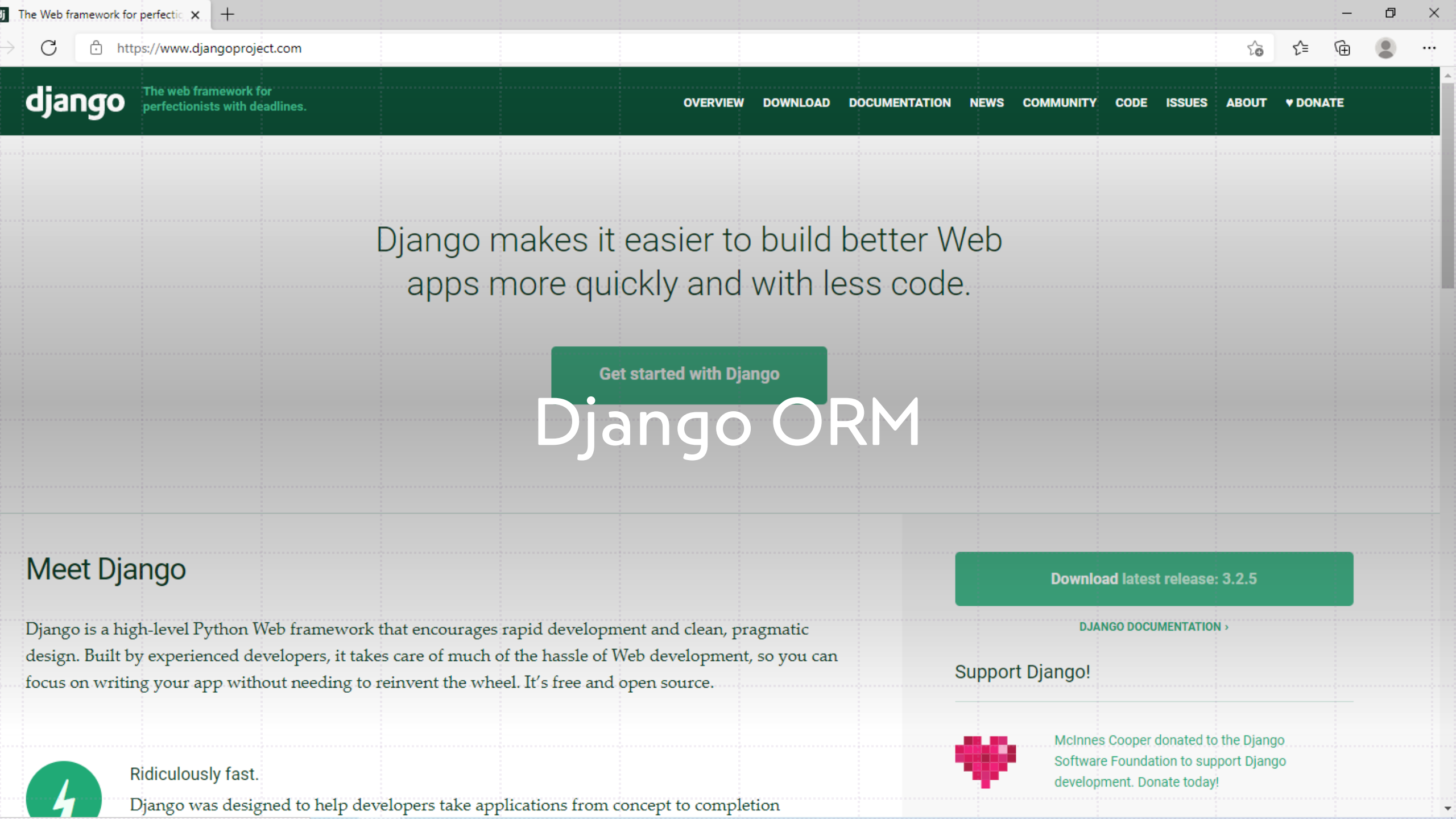
Python

Programowanie baz danych

Plan na dzisiaj

- Przegląd narzędzi do pracy z bazami w Pythonie
- Wstęp do SQLAlchemy

Przegląd narzędzi bazodanowych dla Pythona


The Web framework for perfectic
https://www.djangoproject.com
django
The web framework for perfectionists with deadlines.
OVERVIEW DOWNLOAD DOCUMENTATION NEWS COMMUNITY CODE ISSUES ABOUT ♥ DONATE
Django makes it easier to build better Web apps more quickly and with less code.
Get started with Django
Django ORM
Meet Django
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.
Download latest release: 3.2.5
DJANGO DOCUMENTATION ›
Support Django!
McInnes Cooper donated to the Django Software Foundation to support Django development. Donate today!
Ridiculously fast.
Django was designed to help developers take applications from concept to completion

SQLAlchemy

THE DATABASE TOOLKIT FOR PYTHON

The Python SQL Toolkit and Object Relational Mapper

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

SQLALCHEMY'S PHILOSOPHY

SQL databases behave less like object collections the more size and performance start to matter; object collections behave less like tables and rows the more abstraction starts to matter. SQLAlchemy aims to accommodate both of these principles.

SQLAlchemy

SQLAlchemy considers the database to be a relational algebra engine, not just a collection of tables. Rows can be selected from not only tables but also joins and other select statements; any of these units can be composed into a larger structure. SQLAlchemy's expression language builds on this concept from its core.

SQLAlchemy is most famous for its object-relational mapper (ORM), an optional component that provides the **data mapper pattern**, where classes can be mapped to the database in open ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

SQLAlchemy's overall approach to these problems is entirely different from that of most other SQL / ORM tools, rooted in a so-called **complimentarity-** oriented approach; instead of hiding away SQL and object relational details behind a wall of automation, all processes are **fully exposed** within a series of composable, transparent tools. The library takes on the job of automating redundant tasks while the developer remains in control of how the database is organized and how SQL is constructed.

The main goal of SQLAlchemy is to change the way you think about databases and SQL!

Read some key features of SQLAlchemy, as well as what people are saying about SQLAlchemy.

CURRENT RELEASES

1.4.21 - 2021-07-14 - announce
changes | migration notes | docs

1.3.24 - 2021-03-30 - announce
changes | migration notes | docs

SPONSOR SQLALCHEMY!

Donate Donate to SQLAlchemy through **PayPal**

Sponsor SQLAlchemy through the Tidelift Subscription

LATEST NEWS

SQLAlchemy 1.4.21 Released
Wed, 14 Jul 2021

SQLAlchemy 1.4.20 Released
Mon, 28 Jun 2021

SQLAlchemy 1.4.19 Released

Project Links

Donate to Pallets
Website
PyPI releases
Source Code
Issue Tracker

Contents

Flask-SQLAlchemy
- Requirements
- User Guide
- API Reference
- Additional Information

Quick search

Go

Flask SQLAlchemy

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

See the SQLAlchemy documentation to learn how to work with the ORM in depth. The following documentation is a brief overview of the most common tasks, as well as the features specific to Flask-SQLAlchemy.

Requirements

Our Version	Python	Flask	SQLAlchemy
2.1	2.7, 3.4+	0.12+	0.8+ or 1.0.10+ w/ Python 3.7
3.0+ (in dev)	2.7, 3.5+	1.0+	1.0+

User Guide

- Quickstart
  - A Minimal Application
  - Simple Relationships
  - Road to Enlightenment
- Introduction into Contexts
- Configuration
  - Configuration Keys
  - Connection URI Format
  - Using custom MetaData and naming conventions
  - Timeouts

peewee

Peewee is a simple and small ORM. It has few (but expressive) concepts, making it easy to learn and intuitive to use.

- a small, expressive ORM
- python 2.7+ and 3.4+ (developed with 3.6)
- supports sqlite, mysql, postgresql and cockroachdb
- tons of extensions

Peewee's source code hosted on GitHub.

New to peewee? These may help:

PonyORM - Python ORM with be

https://ponyorm.org

Object-Relational Mapper

Docs Online editor Releases

Pony is a Python ORM with beautiful query syntax

Write your database queries using Python generators & lambdas

Try PonyORM now Support PonyORM

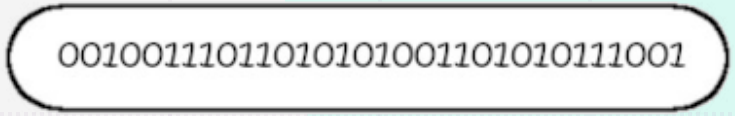
Free open-source software

Github Twitter Telegram Join the newsletter

Custom software development

SQLAlchemy 1.4

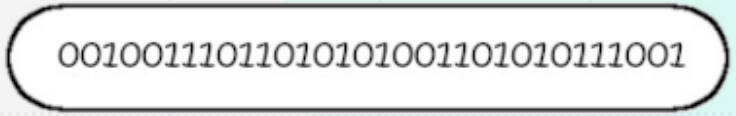
Wstęp


00100111011010101001101010111001

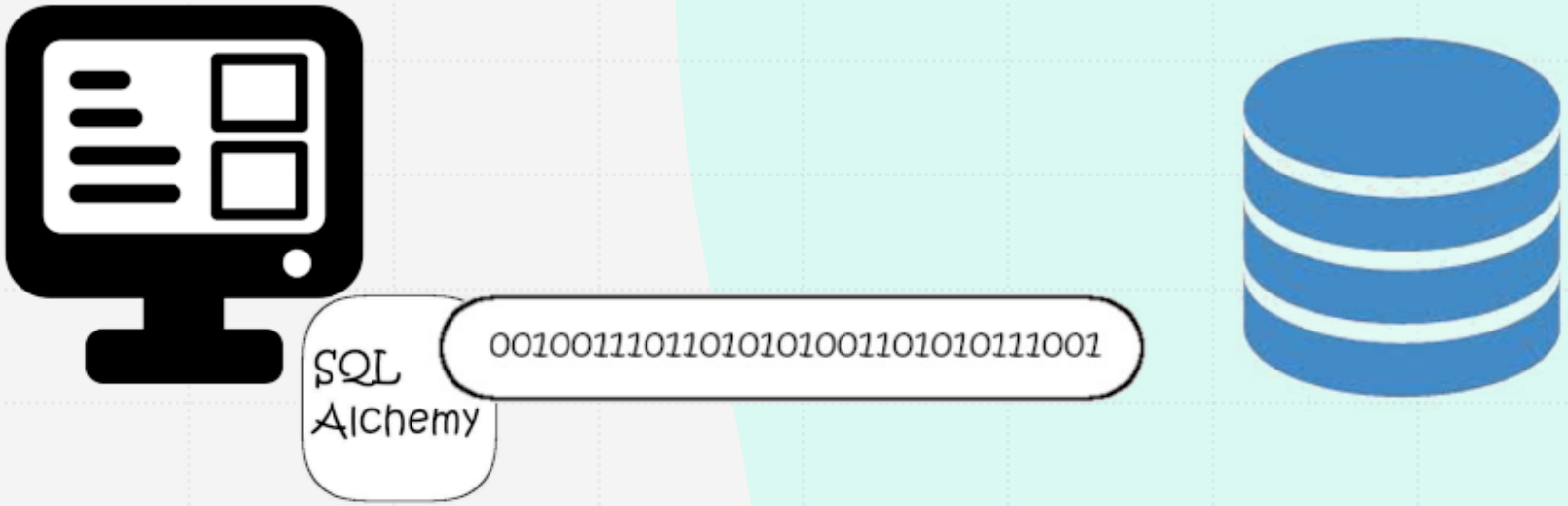
00100111011010101001101010111001

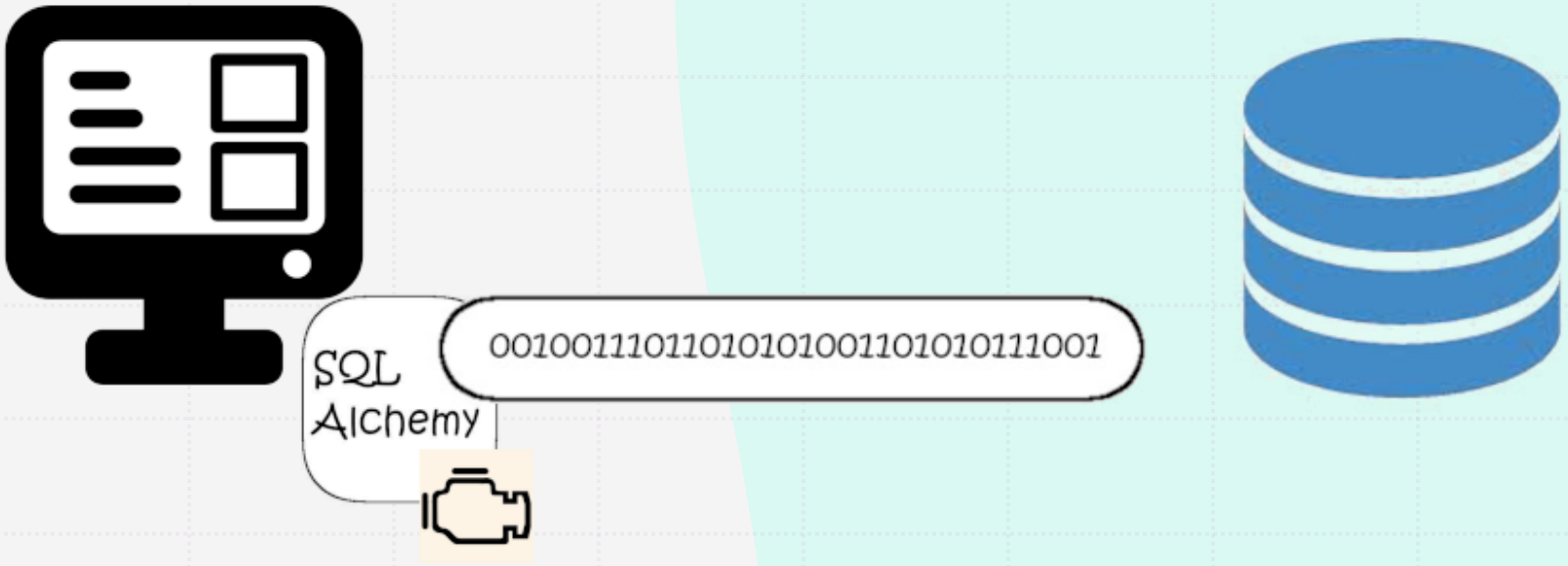
```
CREATE TABLE user
(
  user_id INT NOT NULL PRIMARY KEY,
);
```

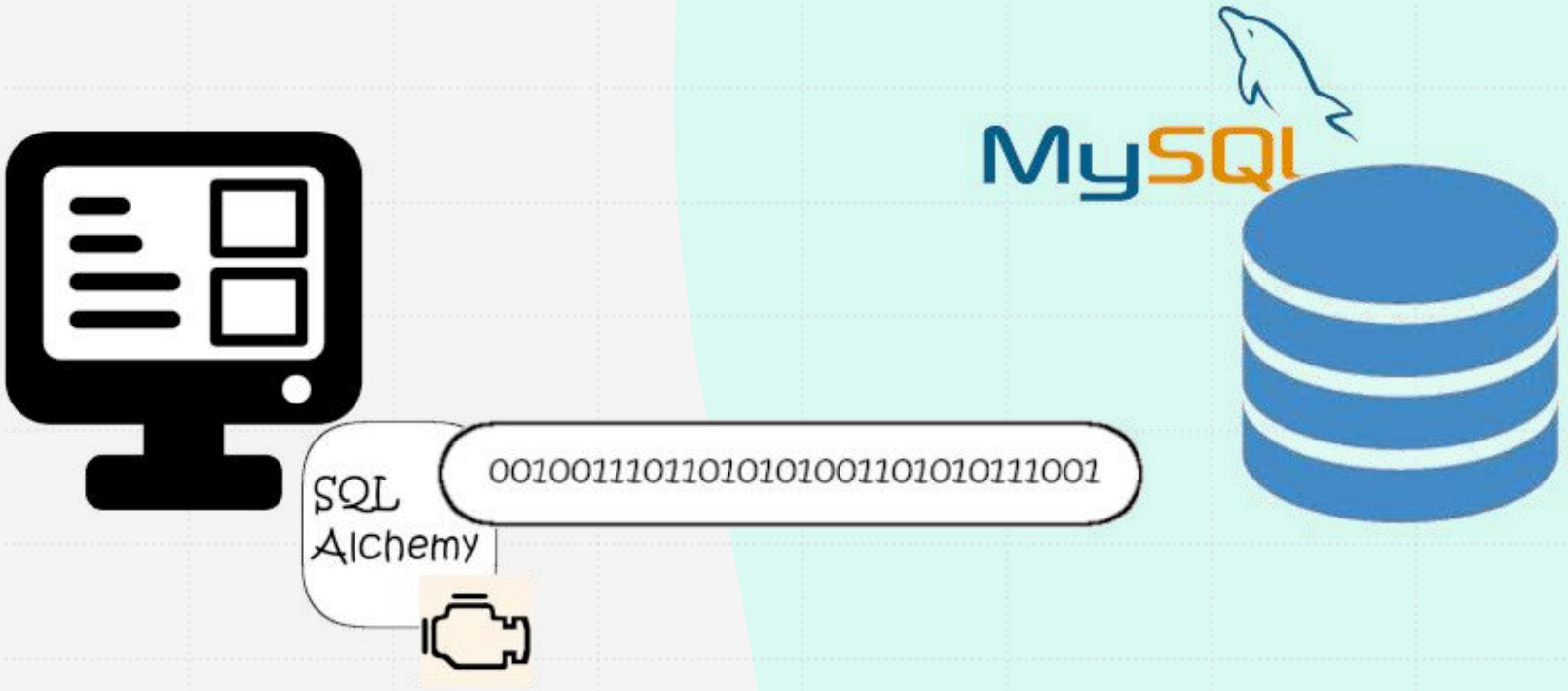
ORACLE



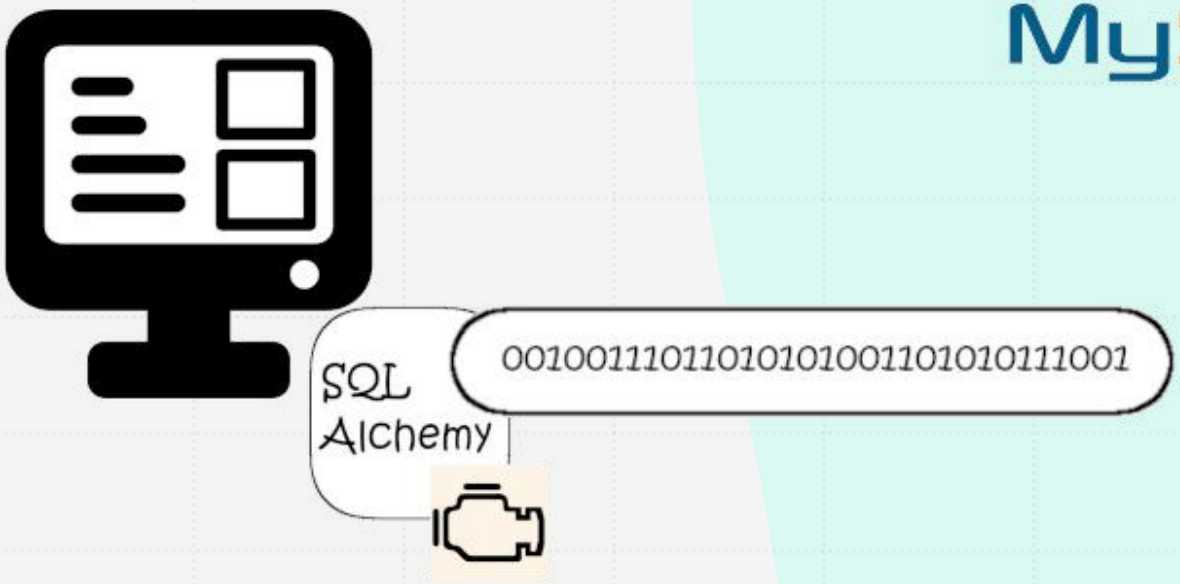
```
CREATE TABLE user
(
  user_id numeric(10) not null,
  CONSTRAINT user_pk PRIMARY KEY (user_id)
);
```


SQL
Alchemy
00100111011010101001101010111001


SQL
Alchemy
00100111011010101001101010111001

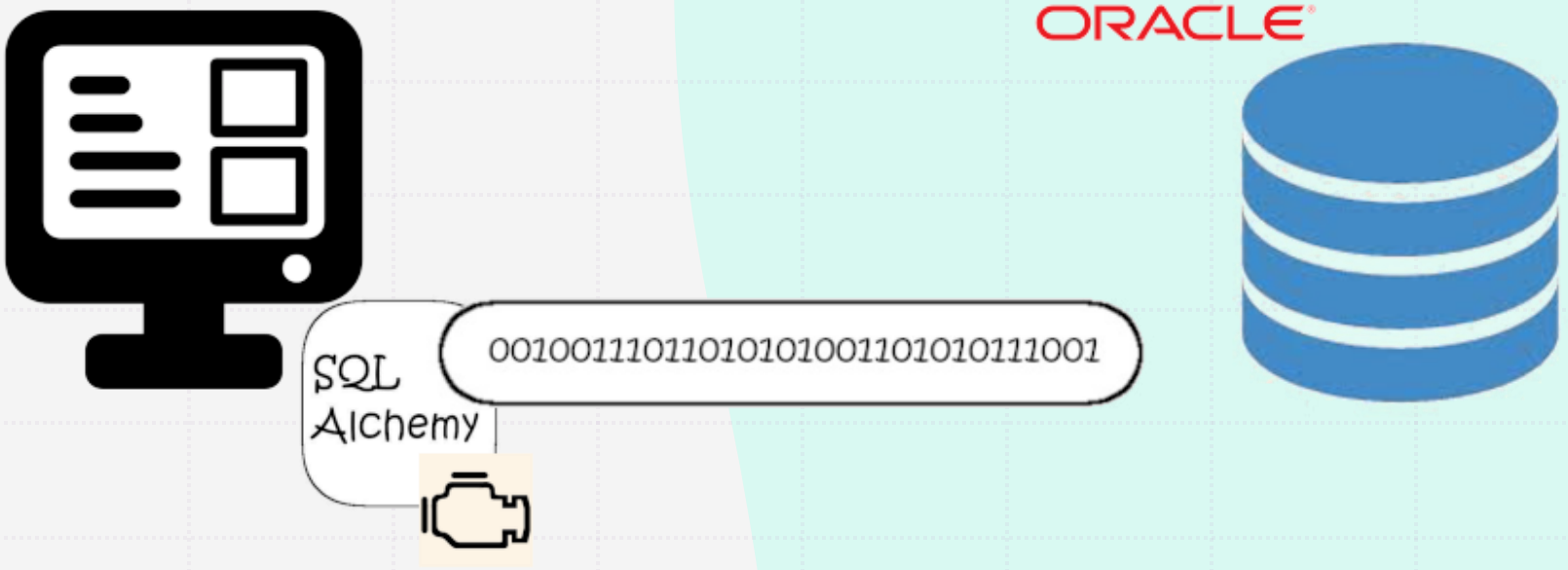



```
Table('user', metadata,
    Column('user_id', Integer, primary_key=True)
)
```

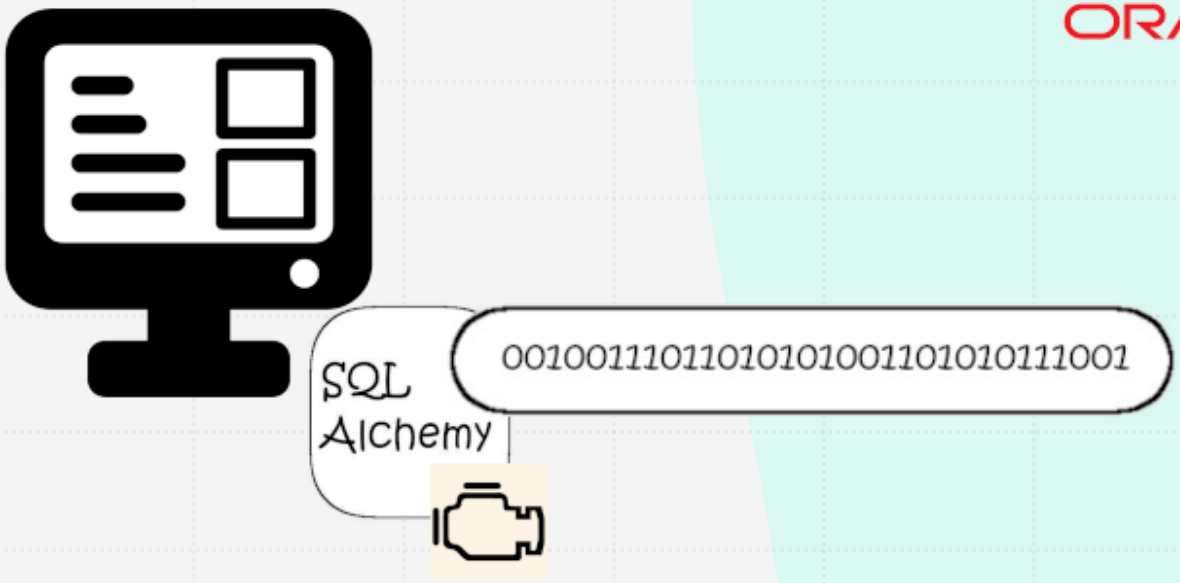


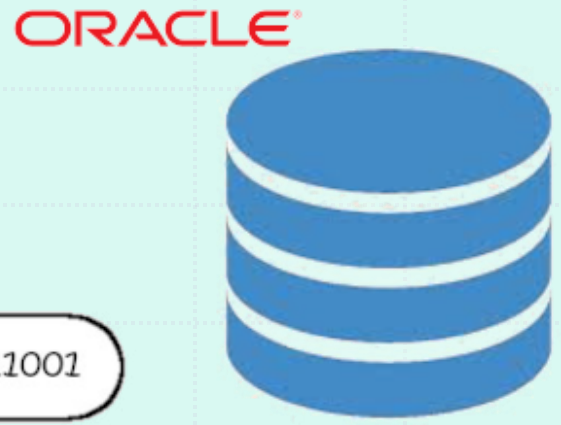

```
Table('user', metadata,
    Column('user_id', Integer, primary_key=True)
)
```

```
CREATE TABLE user
(
  user_id INT NOT NULL PRIMARY KEY,
);
```



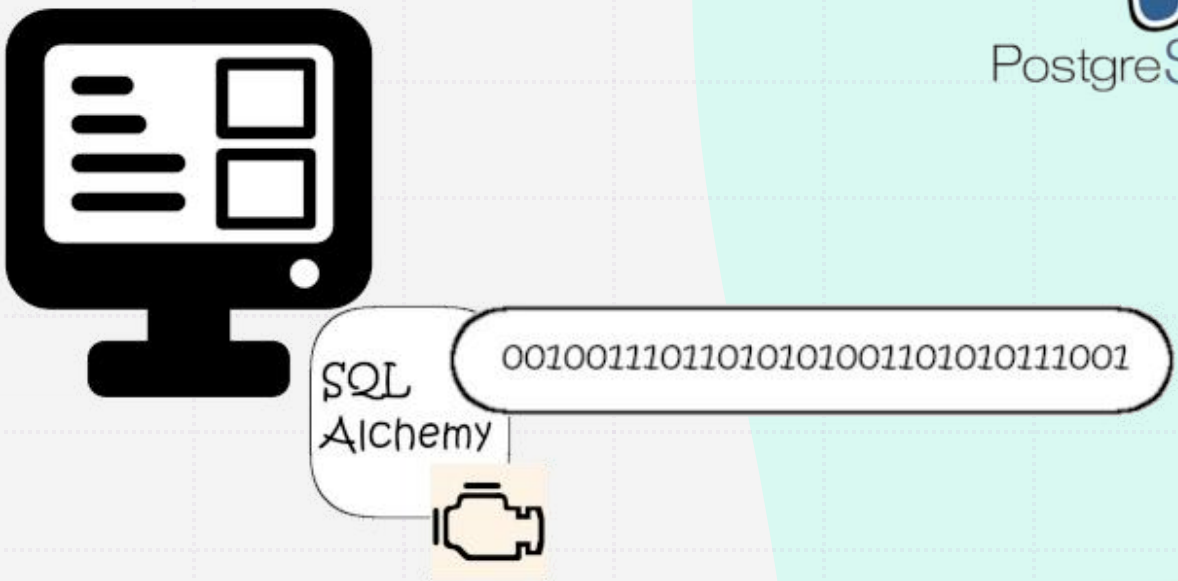

```
Table('user', metadata,
    Column('user_id', Integer, primary_key=True)
)
```






```
Table('user', metadata,
    Column('user_id', Integer, primary_key=True)
)
```

```
CREATE TABLE user
(
  user_id numeric(10) not null,
  CONSTRAINT user_pk PRIMARY KEY (user_id)
);
```




```
Table('user', metadata,
    Column('user_id', Integer, primary_key=True)
)
```

SQLAlchemy 1.4 Documentation

Release: **1.4.25** CURRENT RELEASE | Release Date: September 22, 2021

Contents | Index | Download as ZIP file

Search terms: search... Search

Getting Started

A high level view and getting set up.

Overview | Installation Guide | Frequently Asked Questions | Migration from 1.3 | Glossary | Error Messages | Changelog catalog

Tutorials

SQLAlchemy 1.4 / 2.0 Transitional

SQLAlchemy 2.0 is functionally available as part of SQLAlchemy 1.4, and integrates Core and ORM working styles more closely than ever. The new tutorial introduces both concepts in parallel. New users and those starting new projects should start here!

- SQLAlchemy 1.4 / 2.0 Tutorial - SQLAlchemy 2.0's main tutorial
- Migrating to SQLAlchemy 2.0 - Complete background on migrating from 1.3 or 1.4 to 2.0

SQLAlchemy 1.x Releases

The 1.x Object Relational Tutorial and Core Tutorial are the legacy tutorials that should be consulted for existing SQLAlchemy codebases.

- Object Relational Tutorial (1.x API)
- SQL Expression Language Tutorial (1.x API)

Reference Documentation

SQLAlchemy ORM

- **ORM Configuration:** Mapper Configuration | Relationship Configuration
- **ORM Usage:** Session Usage and Guidelines | Querying Data, Loading Objects | AsyncIO Support
- **Configuration Extensions:** Mypy integration | Association Proxy | Hybrid Attributes | Automap | Mutable Scalars | All extensions
- **Extending the ORM:** ORM Events and Internals
- **Other:** Introduction to Examples

SQLAlchemy Core

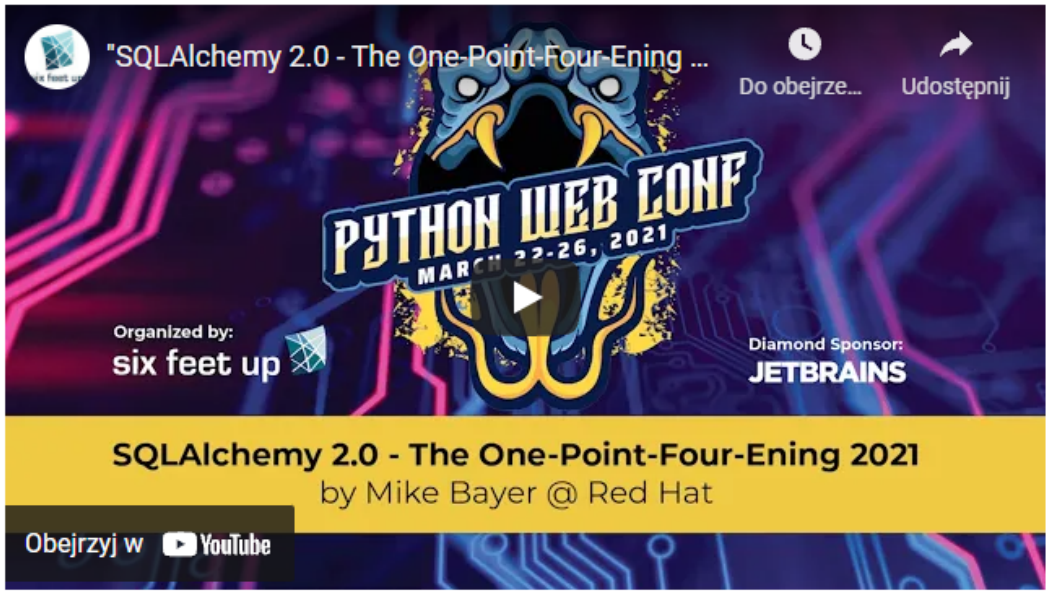
- **Engines, Connections, Pools:** Engine Configuration | Connections, Transactions, Results | AsyncIO Support | Connection Pooling
- **Schema Definition:** Overview | Tables and Columns | Database Introspection (Reflection) | Insert/Update Defaults | Constraints and Indexes | Using Data Definition Language (DD
- **SQL Referenc**

TUTORIALS

The most up-to-date and complete tutorials available for getting started with SQLAlchemy are: * the SQLAlchemy 1.4/2.0 Tutorial which is a full rewrite of the classic "1.x" SQLAlchemy tutorials; users starting with the latest SQLAlchemy releases should start here. * the Core and ORM tutorials are recommended for those using "1.x style" codebases. A few other online resources include:

- ***SQLAlchemy 2.0 - The One-Point-Four-Ening 2021*** **- Python Web Conf 2021**

  Author: Mike Bayer



  This is the newest version of the "getting started" tutorial that presents SQLAlchemy from the perspective of the new 2.0 series.

  - Video
  - Student Download

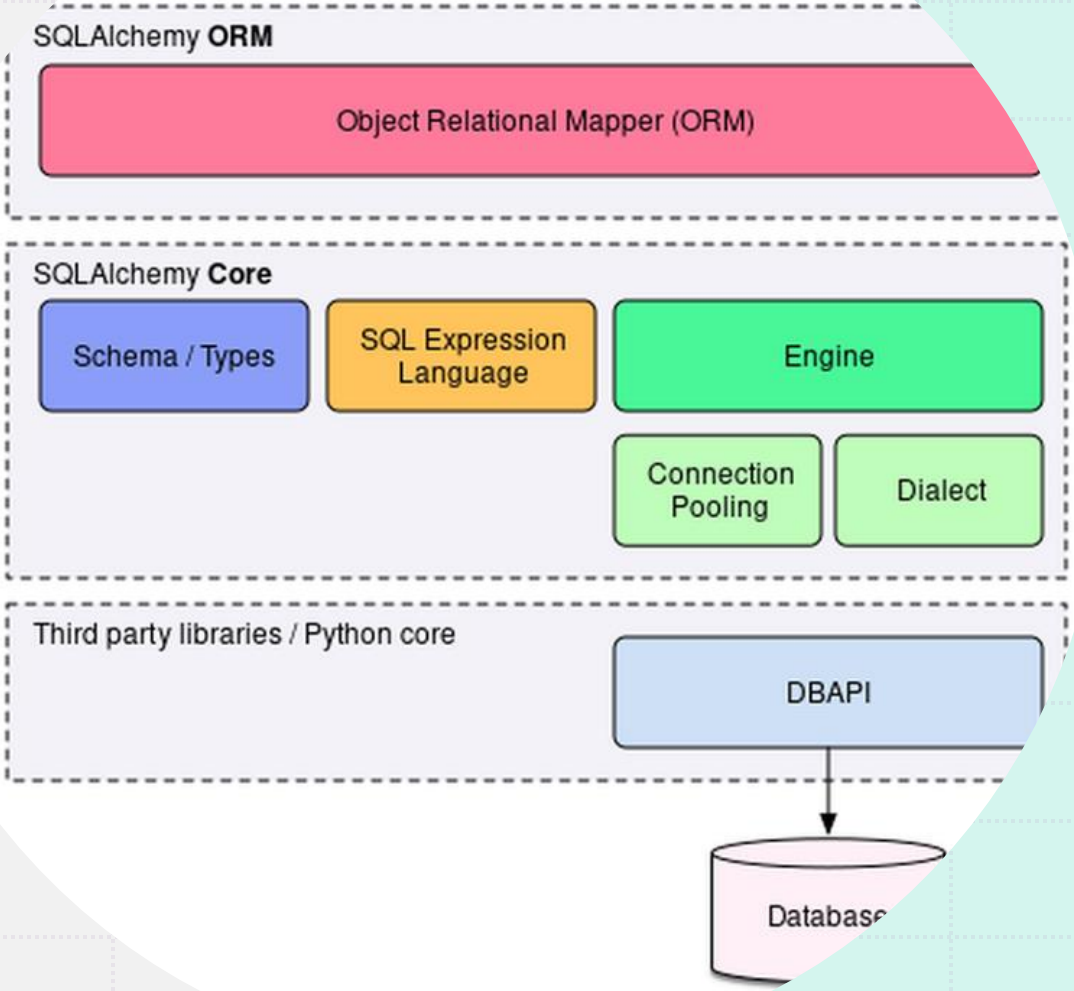
- ***Introduction to SQLAlchemy*** **- presented at many Pycon and other conferences**

Architektura

Dwa podstawowe modele użycia

1. **Core** aka SQL Expression Language
2. **ORM** (Object Relational Mapper)
    a. Mapowanie klasyczne
    b. Mapowanie deklaratywne

Podstawowa architektura



Dialekty

SQLAlchemy 1.3 Documentation

Release: **1.3.12** CURRENT RELEASE | Release Date: December 16, 2

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

SQLite

Support for the SQLite database.

DBAPI Support

The following dialect/DBAPI options are available. Please refer to individual DBAPI sections for connect information.

- pysqlite
- pysqlcipher

Date and Time Types

SQLite does not have built-in DATE, TIME, or DATETIME types, and pysqlite does not
Python *datetime* objects and a SQLite-supported format. SQLAlchemy
when SQLite is used. The implementation classes are
also nicely support ordering. There's no

SQLAlchemy 1.3 Documentation

Release: **1.3.12** CURRENT RELEASE | Release Date: December 16

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

PostgreSQL

Support for the PostgreSQL database.

DBAPI Support

The following dialect/DBAPI options are available. Please refer to individual DBAPI sections for connect information.

- psycopg2
- pg8000
- psycopg2cffi
- py-postgresql
- pygresql
- zxJDBC for Jython

Sequences/SERIAL/IDENTITY

SQLAlchemy 1.3 Documentation

Release: **1.3.12** CURRENT RELEASE | Release Date: December 16, 2

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

MySQL

Support for the MySQL database.

DBAPI Support

The following dialect/DBAPI options are available. Please refer to individual DBAPI sections for connect information.

- mysqlclient (maintained fork of MySQL-Python)
- PyMySQL
- MySQL Connector/Python
- CyMySQL
- OurSQL
- Google Cloud SQL
- PyODBC
- zxjdbc for Jython

Supported Versions

SQLAlchemy 1.3 Documentation

Release: **1.3.12** CURRENT RELEASE | Release Date: December 16

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

Oracle

Support for the Oracle database.

DBAPI Support

The following dialect/DBAPI options are available. Please refer to individual DBAPI sections for connect information.

- cx-Oracle
- zxJDBC for Jython

Connect Arguments

The dialect supports several `create_engine()` arguments which affect the be

- `use_ansi` - Use ANSI JOIN constructs (see the se
- `optimize_limits` - defaults to

SQLAlchemy 1.3 Documentation

Release: **1.3.12** CURRENT RELEASE | Release Date: December 16

SQLAlchemy 1.3 Documentation

CURRENT RELEASE

Contents | Index

Search terms: search...

Microsoft SQL Server

Support for the Microsoft SQL Server database.

DBAPI Support

The following dialect/DBAPI options are available. Please refer to individual DBAPI sections for connect information.

- PyODBC
- mxODBC
- pymssql
- zxJDBC for Jython
- adodbapi

Auto Increment Behavior / IDENTITY Col

SQL Server provides so-called “auto increm

Napis połączeniowy

Definicja

Napis połączeniowy to napis zawierający wszystkie niezbędne informacje do połączenia z bazą danych. Wśród informacji znajdują się m.in:

- *dialekt bazy (sqlite, postgresql, mysql, …)*
- *konektor (ang. connector aka driver) - biblioteka Pythona, która pozwala na pracę z konkretnym dialektem*
- *lokalizacja i nazwa bazy*

Anatomia

Postać uproszczona

```
dialekt:///nazwa_bazy
```

Postać pełna

```
dialect[+driver]://user:password@hostname/dbname[?key=value]
```

Przykłady

SQLite

```
'sqlite:///db_name.db
```

PostgreSQL

```
'postgresql://xavier:postgres@localhost:5432/db_name'
```

MySQL

```
'mysql+mysqlconnector://root:mysql@localhost:3306/db_name'
```

Silnik

Opis

Silnik zapewnia połączenie z bazą oraz obsługuje operacje wykonywane na bazie.

Połączenie z bazą wymaga przekazania do silnika napisu połączeniowego.

Przykład inicjalizacji

```
from sqlalchemy import create_engine

conn_string = 'sqlite:///chinook.sqlite'
engine = create_engine(conn_string)
```

Core

Zadania

Dodatki

SQLite

Small. Fast. Reliable.
Choose any three.

What Is SQLite?

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. More Information...

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems [1] [2] [3] and as a long-term archival format for data [4]. There are over 1 trillion (1e12) SQLite databases in active use [5].

SQLite source code is in the public-domain and is free to everyone to use for any purpose.

Latest Release

Version 3.36.0 (2021-06-18). Download Prior Releases

Common Links

Ongoing development and support of SQLite is made possible in part by SQLite Consortium members, including:

Bloomberg® Expensify Navigation Data Standard

SQLite Tutorial

This **SQLite tutorial** teaches you everything you need to know to start using SQLite effectively. In this tutorial, you will learn SQLite step by step through extensive hands-on practices.

This SQLite tutorial is designed for developers who want to use SQLite as the back-end database or to use SQLite to manage structured data in applications including desktop, web, and mobile apps.

SQLite is an open-source, zero-configuration, self-contained, stand-alone, transaction relational database engine designed to be embedded into an application.

Getting started with SQLite

You should go through this section if this is the first time you have worked with SQLite. Follow these 4-easy steps to get started with SQLite fast.

- First, help you answer the first and important question: what is SQLite? You will have a brief overview of SQLite.
- Second, show you step by step how to download and install the SQLite tools on your computer.
- Third, introduce you to an SQLite sample database and walk you through the steps of using the sample database for practicing.
- Finally, guide you on how to use the sqlite3 commands.

Basic SQLite tutorial

This section presents basic SQL statements that you can use with SQLite. You will first start querying data from the sample database. If you are already familiar with SQL, you will notice the differences between SQL standard and SQL dialect used in SQLite.

Section 1. Simple query

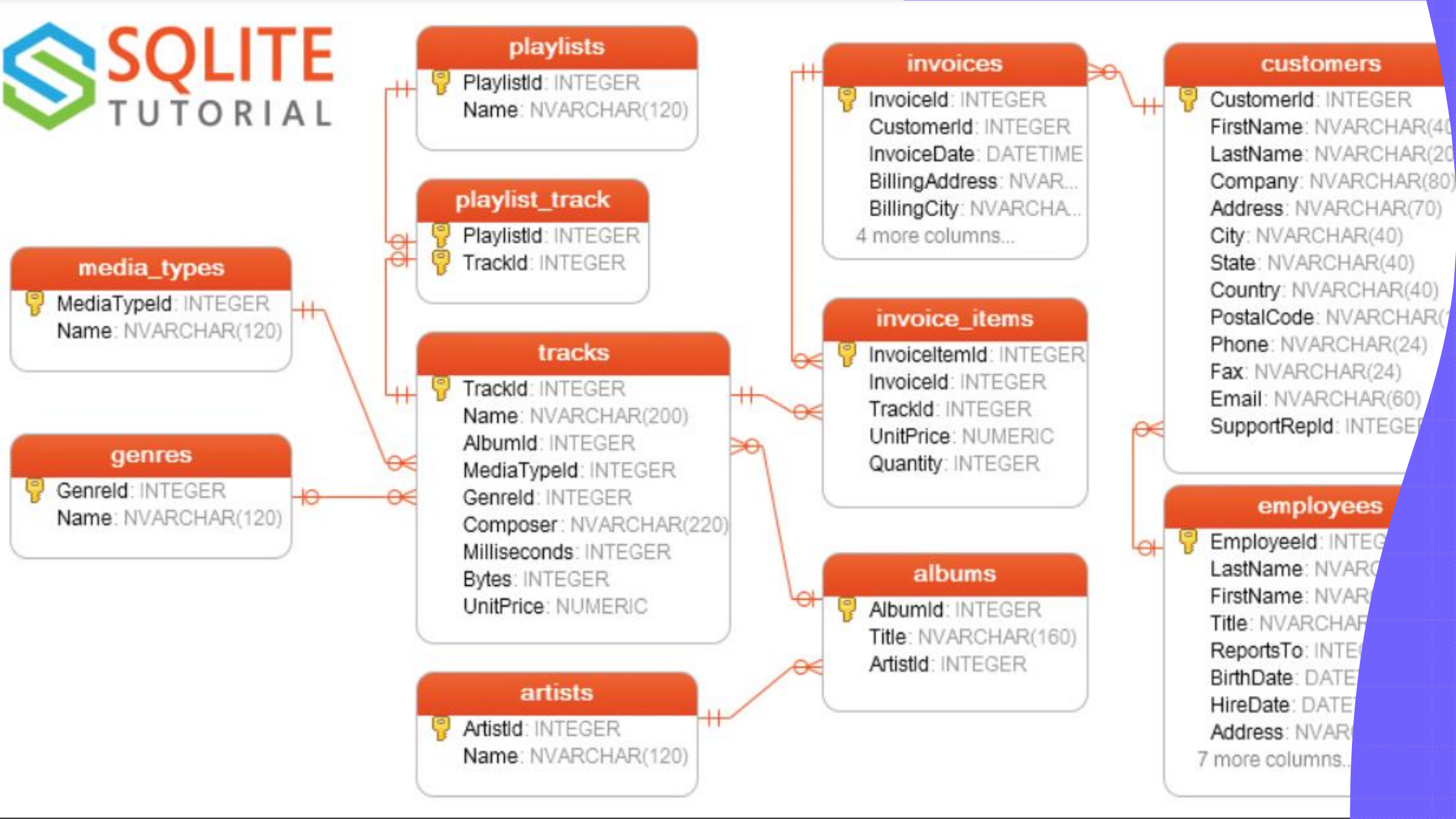
Search ...

GETTING STARTED

What Is SQLite
Download & Install SQLite
SQLite Sample Database
SQLite Commands

SQLITE TUTORIAL

SQLite Select
SQLite Order By
SQLite Select Distinct
SQLite Where
SQLite Limit


SQLITE
TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40
LastName: NVARCHAR(20
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGE
employees
EmployeeId: INTEG
LastName: NVARC
FirstName: NVAR
Title: NVARCHAR
ReportsTo: INTE
BirthDate: DATE
HireDate: DATE
Address: NVAR
7 more columns...