

1. Самый низкоуровневый элемент – DBAPI – драйвер (движок) для работы с БД (открывает соединение, отправляет запросы в БД и работает с ней).

Т.к мы рассматриваем SQLite, то у нас встроенный драйвер sqlite3, который входит в стандартную библиотеку в пайтоне (не надо скачивать). Для postgreSQL несколько драйверов (psycopg2, AsyncPG для асинхронного приложения), но по умолчанию используется psycopg2.

**НО** SQLAlchemy абстрагирует эти драйверы, т.е наважно с какой Бд и драйвером вы работаете, с SQLAlchemy вы создаёте так называемые модели, которые можно «конвертировать» в любую Бд (т.е проводить миграции) .  
Перейдём к наиболее важным высокоуровневым элементам: на схеме объединены пунктиром в ORM и Core.

1. **Core (или ядро)** содержит собственный язык SQL запросов, осуществляет поддержку различных диалектов SQL, а также осуществляет общую интеграцию с самой базой данных, в том числе контролирует соединения с ней.

* SQLAlchemy сама выберет типы данных (ставим integer, а она сама выберет, какой тип данных лучше);
* сама переведёт из пайтон кода в sql-запросы;
* сама открывает-закрывает соединения с БД (фабрика соединений);
* позволяет программисту не задумываться над особенностями той или иной БД (– инкапсуляция)

1. ORM – объектно-реляционная схема – позволяет представлять какие-то данные из БД не в виде строк, а в виде конкретных python-объектов. Python-объектов пренадлежат классам (наз. моделям) – это таблицы в БД. Преобразование объектов, моделей в таблицы и строки Бд (и наоборот) и есть одна из главных задач Алхимии .

Как бы вы ни работали с алхимией: только используя Core или ORM - язык SQL выражений всегда будет представлять собой схематичное отображение базы данных. Разница лишь в том, что в Core он ориентирован на операции, а в ORM на объекты.

II Машина соединений:  
Начало любого приложения с использованием SQLAlchemy начинается с объекта, называемого **машиной соединений** (англ. Engine). Этот объект представляет собой центральный компонент, осуществлящий соединения к отдельным базам данных, предоставляя также фабрику соединений (или пул соединений, англ. connection pool) для работы с ними. Машина является фактически глобальным объектом, создаваемым единожды с заданными настройками для подключения к базе данных, которые чаще всего представляют собой строчку URL.

Для данного туториала мы будем использовать SQLite базу данных, так как это простой способ показать и протестировать простые вещи, без необходимости поднятия полноценного сервера. Машина соединения создается при помощи функции create\_engine(), при этом если передать в качестве ее параметра future=True, то мы получим полный доступ к использованию нового стиля SQLAlchemy 2.0:

Главный аргумент для функции create\_engine это строчка URL, выше мы передали строчку для подключения к SQLite базе данных прямо в оперативной памяти нашего компьютера (то есть даже без создания файла бд). Эта строчка передает три важных факта:

1. С каким типом СУБД мы будем иметь дело? В данном случае это sqlite, который передает SQLAlchemy информацию о том, что необходимо использовать соответствующий **диалект**.
2. Какой DBAPI мы будем использовать? Python DBAPI это сторонний драйвер, который позволяет алхимии взаимодействовать с отдельными базами данных. В данном случае мы используем pysqlite, который в Python3 заменен на sqlite3, входящим в стандартную библиотеку. Если это не будет указано, то SQLAlchemy сам решит какой DBAPI использовать.
3. И в конце концов, где мы собираемся эту базу данных распологать. В данном случае URL включает фразу /:memory:, которая говорит sqlite3, что мы будем работать с in-memory-only базой данных

Обратите внимание, что мы также передали параметр echo=True, который приказывает нашей машине сообщать о всех SQL запросах в логер Python, который в свою очередь будет передавать их в системный вывод (условно консоль). По большому счету, данный параметр всего лишь более простой путь настроить логирование и это может быть очень полезно в некоторых ситуациях..

ИТОГО: строка URL, переданная в функцию, определяет тип СУБД (sqlite), используемый DBAPI (pysqlite) и расположение базы данных (in-memory). Параметр echo=True активирует логирование SQL-запросов для удобства отладки.

Параметр future=True в create\_engine указывает на использование будущего API SQLAlchemy. Это означает, что используются более современные и потенциально измененные функции и механизмы работы с ORM и SQLAlchemy, что может улучшить производительность и совместимость с будущими версиями. Включение этого параметра позволяет разработчикам готовиться к изменениям в API, которые могут произойти в будущих релизах.

Сессия в SQLAlchemy — это объект, который управляет операциями с базой данных, включая транзакции. Она обеспечивает интерфейс для выполнения запросов и сохранения изменений.

Сессия может содержать одну или несколько транзакций и отслеживает изменения объектов. Когда вы вызываете commit(), все изменения в текущей сессии фиксируются в базе данных. Если возникают ошибки, вы можете вызвать rollback(), чтобы отменить все изменения в рамках сессии.