Y_LAB DEVELOPMENT

Интенсив Python

Вебинар №1:
Базы данных, ORM, Фреймворки

О компании

YLab Development

- Аутсорс-продакшн компания
- Реализация IT-проектов
- Автоматизация бизнес-процессов

Официальные сайты:

- https://ylab.io
- https://university.ylab.io/
- https://t.me/ylab_v_it
- https://www.youtube.com/@ylabuniversity

О курсе

Цель курса

- Получение опыта разработки ПО на Python
- Формирование портфолио
- Знакомство с современным стеком технологий

Оценка

- Накопительная по итогу выполнения ДЗ
- Экзамена нет
- Личное интервью перед набором на стажировку

Объем

- Лекции– 4
- Домашних заданий – 4

Y_LAB DEVELOPMENT

Требования к слушателям

Программирование

- Владение базовым синтаксисом Python
- Знание ООП

Сетевое взаимодействие

- Понимание как работает общение между клиентом и сервером

Y_LAB DEVELOPMENT

Темы

- Базы данных
- ORM
- Python фреймворки

Y_LAB DEVELOPMENT

Часть 1. Базы данных

База данных — это организованная структура, предназначенная для хранения, изменения, поиска, анализа и обработки взаимосвязанной информации, преимущественно больших объемов.

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними:

- по модели;
- по распределенности.

Y_LAB DEVELOPMENT

Виды баз данных:

Реляционные

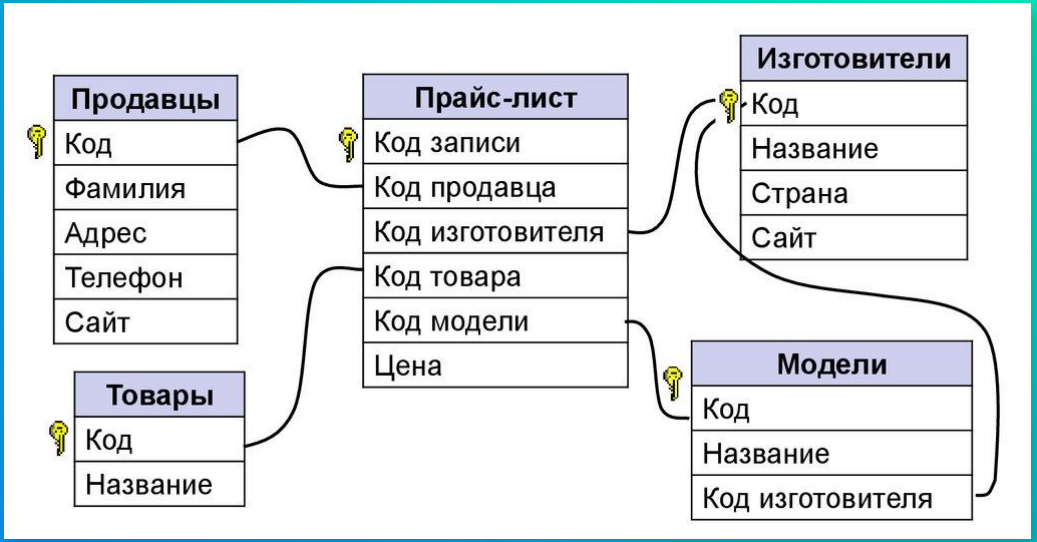
- MySQL
- PostgreSQL
- Oracle Database
- Microsoft SQL Server

NoSQL

- MongoDB
- Apache Cassandra
- CouchDB
- Apache HBase
- Redis

Реляционные БД

Реляционная база данных - это набор простых таблиц между которыми установлены связи



Часть 1. Базы данных

По типам связь между объектами классифицируются на следующие:

- один к одному

(Водитель Яндекс такси - Служебный автомобиль)

- один ко многим, многие к одному

(Новостной пост в социальной сети - комментарии к посту)

- многие ко многим

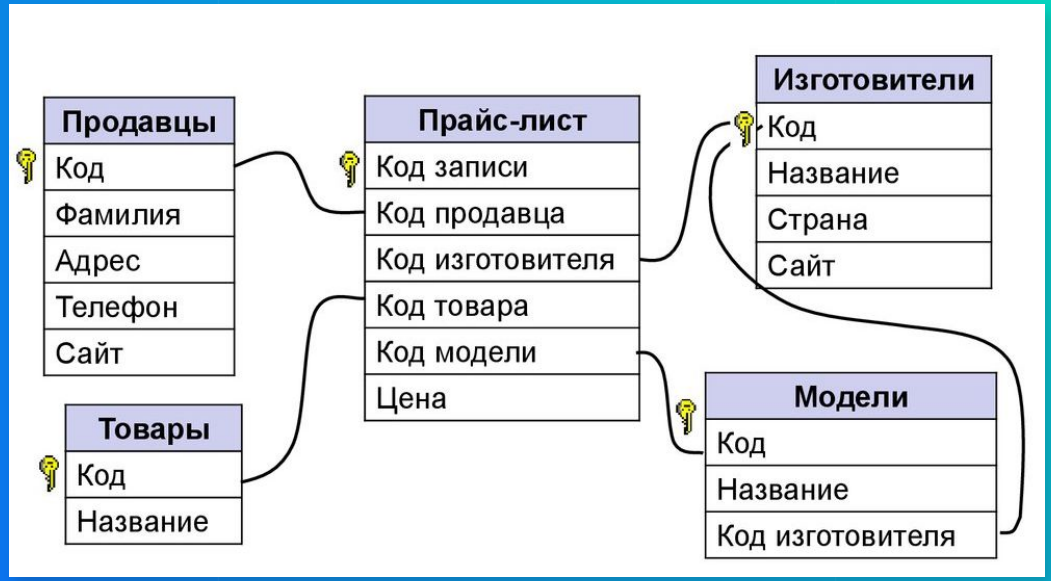
(Автор - Книги)

Y_LAB DEVELOPMENT

Нормализация базы данных

Избегание дубликатов

Ускорение запросов



Логическая изоляция данных

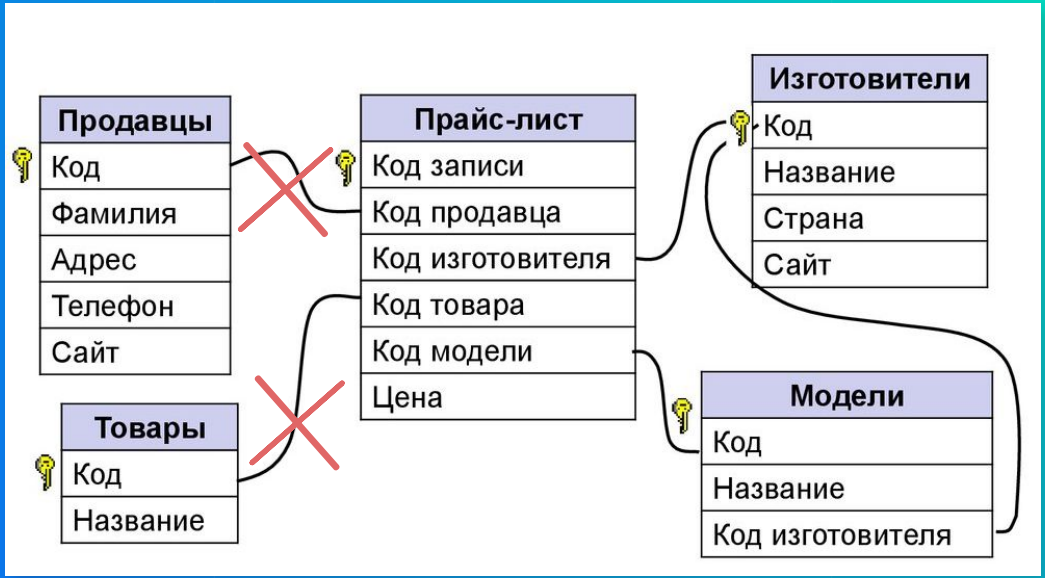
Согласованность данных

Денормализация

~~Избегание~~ дубликатов

Повышение размера БД

Ускорение запросов



~~Логическая изоляция данных~~

~~Согласованность данных~~

Денормализация – намеренное приведение структуры БД в состояние, не соответствующее критериям нормализации.

Y_LAB DEVELOPMENT

Реляционные БД

Плюсы:

- Универсальность: могут хранить практически любую информацию.
- Язык запросов: SQL, язык запросов, используемый в РБД, мощный и гибкий.
- Масштабируемость: РБД можно масштабировать вверх, увеличивая мощность единственной машины.
- Стандарт ACID: РБД следуют принципам ACID (атомарность, согласованность, изолированность, долговечность), что обеспечивает надежность транзакций.

Y_LAB DEVELOPMENT

Реляционные БД

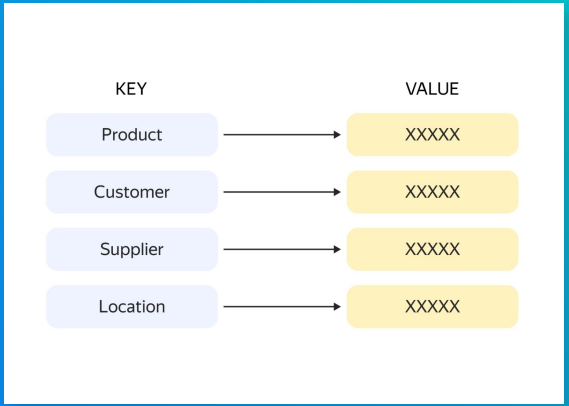
Минусы:

- Низкая гибкость: РБД требуют заранее определенной схемы данных и могут быть не так гибко настраиваемыми, как NoSQL.
- Время на проектирование: разработка схемы и структуры таблиц в РБД может занять много времени, особенно для сложных систем.
- Производительность: для некоторых типов запросов и операций (например, запросов, включающих сложные связи между таблицами) РБД могут быть медленнее
- Сложность: SQL и РБД могут быть сложными для изучения и использования для новичков.

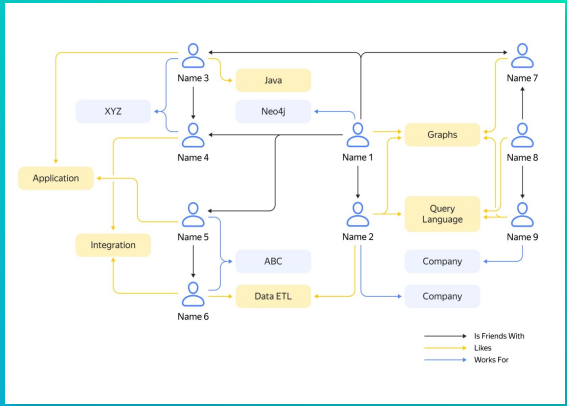
Часть 1. Базы данных

NoSQL

Document-oriented



Key-value store



Graph store

Y_LAB DEVELOPMENT

NoSQL базы данных

Плюсы:

- Масштабируемость: NoSQL базы данных легко масштабируются горизонтально.
- Скорость: NoSQL базы данных обычно быстрее, чем реляционные базы данных для больших объемов данных.
- Гибкость: NoSQL базы данных поддерживают гибкую структуру данных. Вам не нужно определять структуру, такую как столбцы и типы данных, заранее.
- Скорость разработки: нет необходимости проектировать структуру БД, в отличие от реляционных БД.

Y_LAB DEVELOPMENT

NoSQL базы данных

Минусы:

- Отсутствие стандарта:
  Разные базы данных NoSQL используют разные интерфейсы и языки запросов, что может создать сложности при смене или взаимодействии между разными моделями NoSQL.

- Сложность транзакций:
  Большинство баз данных NoSQL не поддерживают ACID транзакции (атомарность, согласованность, изолированность, продолжительность) на разных узлах.

- Неполная поддержка возможностей запросов:
  Ряд NoSQL-баз данных не поддерживает некоторые функции, такие как внешние соединения или агрегационные функции, которые доступны в реляционных базах данных.

Y_LAB DEVELOPMENT

СУБД

СУБД - это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

Наиболее распространенными СУБД: MySQL, PostgreSQL, Oracle, Microsoft

PostgreSQL

MySQL®

Microsoft® SQL Server®

Y_LAB DEVELOPMENT

SQL

```sql
SELECT `phone_book` FROM `test` ORDER BY `surname`;

INSERT INTO `phone_book` (`id`, `surname`, `name`, `patronymic`, `phone`)
VALUES ('1', 'Абрамов', 'Максим', 'Викторович', '555-55-55');

UPDATE `phone_book`
SET `surname` = 'Абрамович'
WHERE `id`=1;

DELETE FROM `phone_book`
WHERE `id`=1;
```

SQL – это стандарт который применяется для создания, модификации и управления объектами и данными в реляционной базе данных.

Y_LAB DEVELOPMENT

SQL

Пример сложного SQL запроса

```
SELECT
      r.NAME AS "NAME"
       ,ol.DATETIME AS "DATETIME"
      ,ol.PARAMETER AS "PARAMETER"
      ,e.NAME AS "OPERATOR"
      ,e2.NAME AS "MANAGER"
      ,COUNT(ORDERS00.ORDERNAME) as "CNTORDERS"
      ,r.SIFR AS "SIFR"
      ,r.GUIDSTRING AS "GUIDSTRING"
  FROM OPERATIONLOG ol
left join VISITS v on v.SIFR=ol.VISIT and v.MIDSERVER=ol.MIDSERVER and v.
left join EMPLOYEES e on e.SIFR = ol.OPERATOR
left join EMPLOYEES e2 on e2.SIFR = ol.MANAGER
left join CASHES c on c.SIFR = ol.STATION
left join CASHGROUPS cg on cg.SIFR = c.CASHGROUP
left join RESTAURANTS r on r.SIFR = cg.RESTAURANT
left join ORDERS ORDERS00
  ON ORDERS00.VISIT=ol.VISIT AND ORDERS00.MIDSERVER=ol.MIDSERVER AND ORDE
WHERE (r.GUIDSTRING IN (:RESTAURANT2)) AND (ol.DATETIME >= :date1) AND (o
 OPERATION=833
group by
       ol.DATETIME
      ,ol.PARAMETER
      ,e.NAME
      ,e2.NAME
      ,r.NAME
      ,r.SIFR
      ,r.GUIDSTRING
order by ol.DATETIME
```

Y_LAB DEVELOPMENT

Часть 2. ORM

- **ORM (Object-Relational Mapping)** — это фреймворк для генерирования SQL на основе объектов модели.
- **ORM** даёт нам неосведомленность об используемой БД: ее суть в том, что наше приложение не должно ничего знать о способах загрузки или хранения данных.

По итогу независимость от конкретных технологий баз данных.

Y_LAB DEVELOPMENT

Паттерны для ORM:

Active record:

- Django ORM
- Tortoise ORM
- Orator

Data mapper:

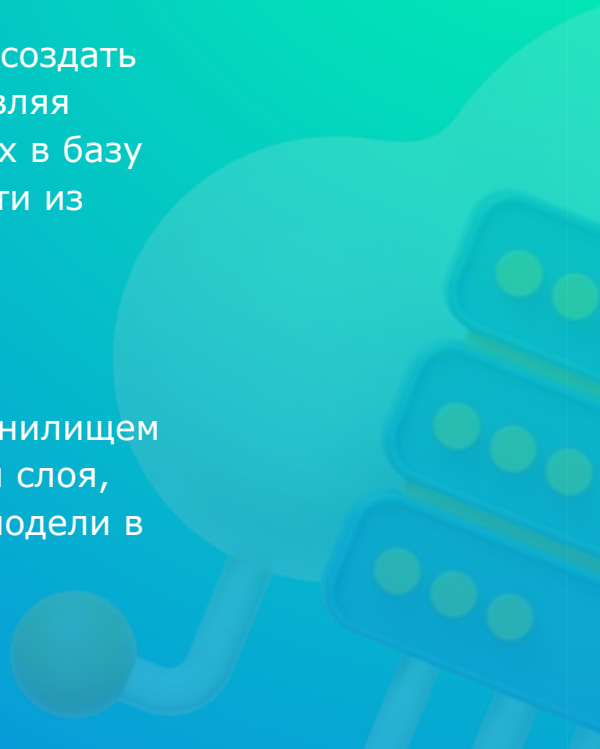
- SQLAlchemy

Y_LAB DEVELOPMENT

Паттерны для ORM:

Active record - это дизайн-шаблон, позволяющий упрощенно создать и использовать сохраняемые в базе данных объекты, осуществляя роль модели в MVC. Тут модель отвечает за сохранение данных в базу данных, что нарушает принцип единственности ответственности из SOLID.

Data Mapper - это слой доступа к данным, обеспечивающий двунаправленное отображение данных между постоянным хранилищем и хранилищем в памяти. Отличается от Active Record наличием слоя, такого как entityManager, отвечающего за перенос состояния модели в базу данных и обратно.



Y_LAB DEVELOPMENT

Паттерны для ORM:

Django

```
1  Football.objects.filter(team__name="Manchester United")
```

SQLAlchemy

```
1  session.query(Football).join(Football, Team).filter(Team.name=="Kamma Sing")
```

Django более абстрактен в своем запросе и показывает только установленное соединение между различными таблицами базы данных, в то время как **SQLAlchemy** идет гораздо глубже.

Y_LAB DEVELOPMENT

Паттерны для ORM:

Django

```python
Student.objects.filter(id=Subquery(
      StudentCourse.objects.filter(
      student=OuterRef('id'),final_grade__gte=10
)[:1].values('student')))
```

Требуется найти студентов, которые получили хотя бы одну 10 балльную оценку, вне зависимости от курса.

SQLAlchemy

```python
db.query(Student).filter(
    Student.id == db.query(StudentCourse.student_id).filter(
        StudentCourse.student_id == Student.id, StudentCourse.final_grade >= 10
    ).limit(1).subquery()
)
```

Y_LAB DEVELOPMENT

Часть 3. Фреймворки

Фреймворк – архитектурный каркас приложения, включающий набор готовых решений, в виде функций, модулей или подсистем. Например, подсистема регистрации и авторизации пользователей.

- Помогают быстро начать разработку программной системы.
- Упрощают взаимодействие с базой данных.
- Реализуют шаблоны проектирования.
- Повышают степень повторного использования кода.
- Обладают документацией и поддержкой сообщества.

Y_LAB DEVELOPMENT

Фреймворк Django

Django:

- Универсальный и мощный фреймворк.
- Обладает подробной документацией и развитым сообществом.
- Имеет свою ORM
- Обладает встроенной панелью управления сайтом.
- В Django присутствует каталог на сотни плагинов.
- Достаточно просто подключаются аналоги.
- «Из коробки»встроены шаблонизатор, мультиязычность, автоматическая документация и так далее.
- Django GitHub: https://github.com/django/django

Y_LAB DEVELOPMENT

Фреймворк Flask

Flask:

- Это скелет, на который разработчик может навесить любой удобный для него инструментарий.
- Выбирая Flask для проекта, разработчик волен в выборе ORM. Как правило, выбор падает на SQLAlchemy.
- Обладает подробной документацией и развитым сообществом.
- Легковесный фреймворк.
- Возможность разработки веб-сайтов с использованием шаблонизатора (Jinja2).

Y_LAB DEVELOPMENT

Фреймворк FastAPI

FastAPI:

- Легкое создание API-серверов со встроенными валидацией, сериализацией и асинхронностью.
- Работой с web в FastAPI занимается фреймворк Starlette.
- За валидацию отвечает библиотека Pydantic.
- Встроенные фоновые задачи и веб-сокеты.
- Авто-документация.

Y_LAB DEVELOPMENT

Спасибо за внимание!

Увидимся в следующих лекциях.

Наш университет: https://university.ylab.io/