АКАДЕМИЯ АЙТИ

День 26

academyit.ru

Типы сортировки. Основные алгоритмы сортировки и поиска. Разработка на Python оптимальных алгоритмов поиска. Оптимизация алгоритма.

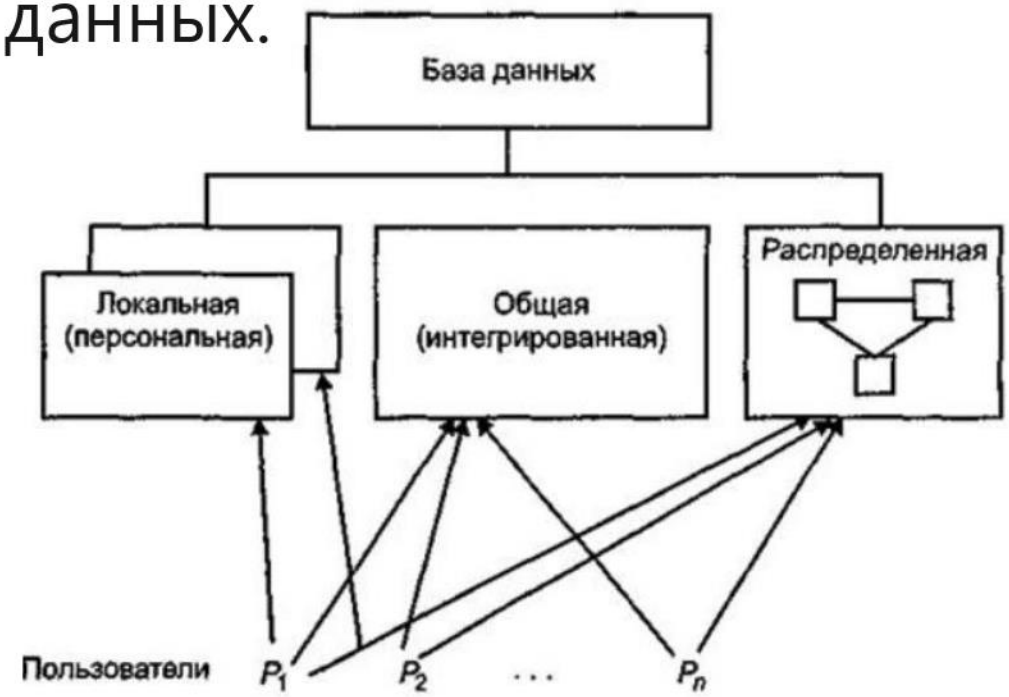
Базы данных. Реляционная модель данных. СУБД SQLite. Язык SQL, основные команды. Работа с БД в Python.

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

OpenServer, phpMyAdmin

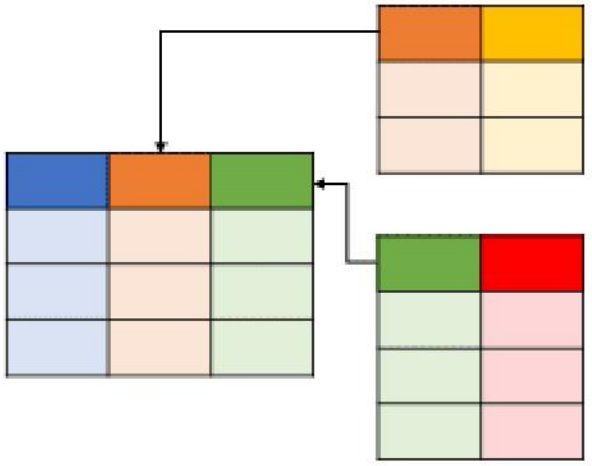
Базы данных

- Базы данных – хранилище структурированной информации.
- Система управления базами данных (СУБД) – информационная система, обеспечивающая эффективный поиск, обновление, добавление и удаление данных.



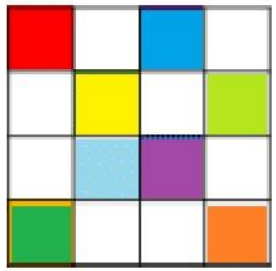
Классификация СУБД по модели данных

SQL

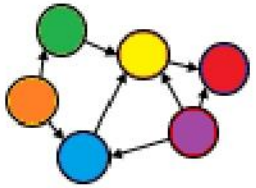


реляционные СУБД

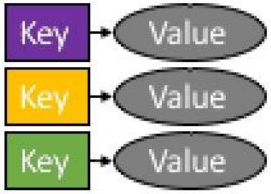
NoSQL



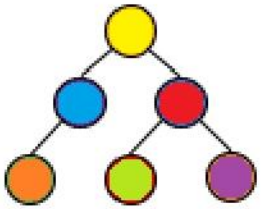
колоночные



графовые



ключ-значение



документо-ориентированные

Реляционная модель данных

- Таблица (отношение, сущность) - набор данных, описывающих определенный тип объектов. Состоит из строк и столбцов.
- Столбец (атрибут, поле) - содержит определенную информацию о каждом объекте (например, FirstName).
- Имеет определенный тип данных (целочисленный, строковый, ….). Может иметь неопределенное значение (null).
- Строка (кортеж, запись) - содержит информацию об одном объекте. Каждая строка в таблице уникальная. Строка содержит первичный ключ.
- Связи между таблицами устанавливаются на основе внешних ключей – когда запись одной таблицы содержит поле, хранящее значение ключей другой таблицы.

Пример. БД Фильмы

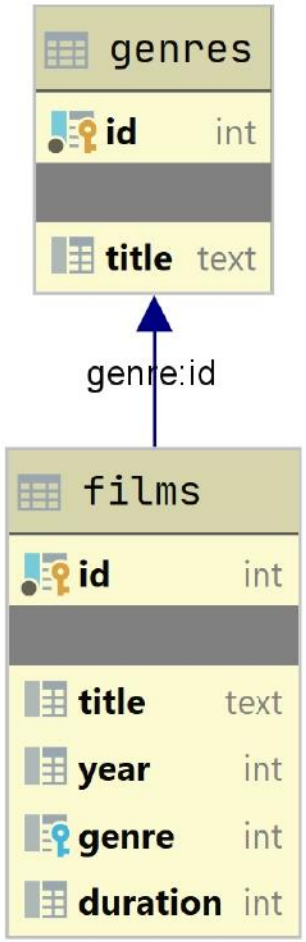


Таблица genres состоит из двух полей: id и title, в которых для каждого жанра хранится его идентификатор и название соответственно.

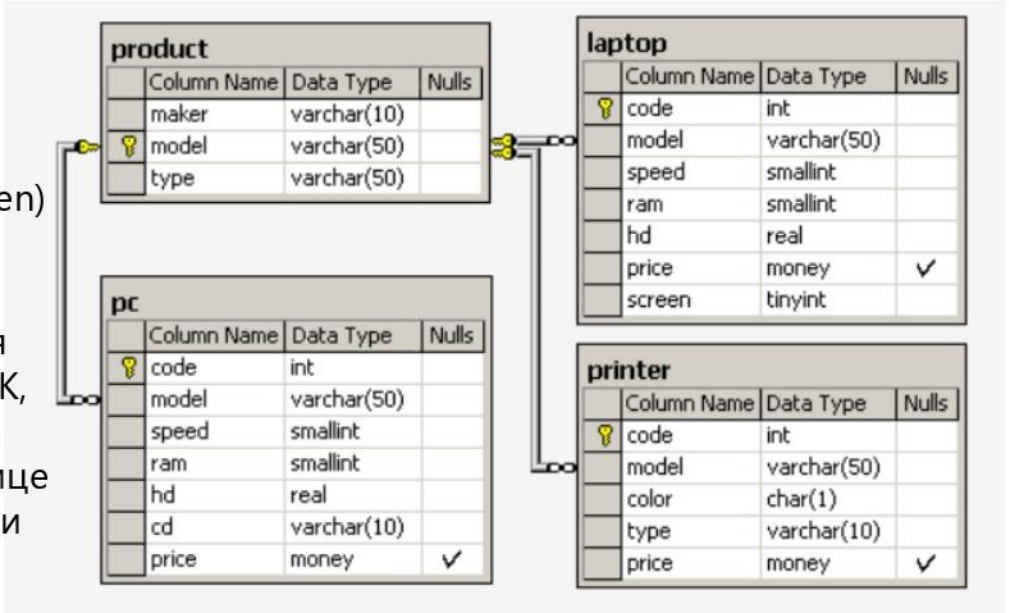
В таблице films есть поля id, title, year, genre, duration для хранения идентификатора фильма, его названия, года выпуска, идентификатора жанра и длительности в минутах.

Кроме того, между таблицами есть связь, которая говорит о том, что номер жанра genre у записи films соответствует записи в таблице genres с таким же значением идентификатора.

Пример. БД Компьютеры

Схема БД состоит из четырех таблиц:
Product(maker, model, type)
PC(code, model, speed, ram, hd, cd, price)
Laptop(code, model, speed, ram, hd, price, screen)
Printer(code, model, color, type, price)

Таблица Product представляет производителя (maker), номер модели (model) и тип ('PC' - ПК, 'Laptop' - ПК-блокнот или 'Printer' - принтер). Предполагается, что номера моделей в таблице Product уникальны для всех производителей и типов продуктов.



В таблице PC для каждого ПК, однозначно определяемого уникальным кодом – code, указаны модель – model (внешний ключ к таблице Product), скорость - speed (процессора в мегагерцах), объем памяти - ram (в мегабайтах), размер диска - hd (в гигабайтах), скорость считывающего устройства - cd (например, '4x') и цена - price.

Таблица Laptop аналогична таблице PC за исключением того, что вместо скорости CD содержит размер экрана -screen (в дюймах).

В таблице Printer для каждой модели принтера указывается, является ли он цветным - color ('y', если цветной), тип принтера - type (лазерный – 'Laser', струйный – 'Jet' или матричный – 'Matrix') и цена - price.

Реляционные СУБД

- Клиент-серверные СУБД
  - Свободно распространяемые MySQL, PostgreSQL
  - Коммерческие Oracle, MS SQL Server
- Мы будем использовать СУБД SQLite. Она разработана как отдельный модуль, встраиваемый в приложение и не требует установки ПО.
- База данных – один файл, который легко перенести на другой компьютер.

Язык SQL

- Для работы с базами данных разработан специальный язык — SQL (structured query language — «язык структурированных запросов»).
- Язык SQL поддерживают все реляционные СУБД и многие нереляционные хранилища данных.
- SQL включает основные команды для работы с базами – SELECT (чтение данных), INSERT(добавление), UPDATE(обновление) и DELETE (удаление).
- Также есть множество других команд для создания таблиц, описания хранимых процедур, обработки транзакций и т.п.

Запрос на выборку данных Select

SELECT <столбцы через запятую или символ * для выбора всех столбцов>

FROM <таблицы через запятую>

[WHERE <условие или фильтр>]

[GROUP BY <столбцы через запятую, по которым нужно сгруппировать данные>]

[HAVING <условие в уже сгруппированных данных>]

[ORDER BY <столбцы через запятую, по которым нужно отсортировать вывод>]

Примеры простых запросов

```
SELECT * FROM Films WHERE year = 2005
```

- фильмы, выпущенные в 2005 году

```
SELECT * FROM Films WHERE year > 2005 AND duration >= 45 AND duration <= 90
```

фильмы, выпущенные после 2005 года
с продолжительностью от 40 минут
до 1,5 часов

Операции в условии WHERE

"=" равенство

"<>" или "!=" неравенство

"<" меньше, "<=" меньше или равно

"!>" не больше, ">" больше

">=" больше или равно или !< не меньше

"BETWEEN" между двумя значениями

"IS NULL" пустое поле

IN NOT IN вхождение/невхождение в список ()

LIKE проверка шаблона (% — обозначает любое количество любых символов, _ — обозначает один любой символ)

AND, OR, NOT логические связки между условиями

Примеры условий отбора

duration IN (45, 90) - значение есть в списке?

duration BETWEEN 45 AND 60 - значение в диапазоне

title like 'A%' – начинается с буквы A, % - любое количество произвольных символов

SELECT DISTINCT year FROM Films – убирает повторы

Подзапросы

```
SELECT title FROM Films
WHERE genre=
( SELECT id FROM genres WHERE title =
'фантастика')

SELECT Films.title, Genres.title
FROM Films, Genres
WHERE Films.genre = Genres.id AND Genr
es.title ='фантастика'
```

Сортировка

SELECT DISTINCT year FROM Films ORDER BY year – сортировка по возрастанию

SELECT DISTINCT year FROM Films ORDER BY year DESC – сортировка по убыванию

SELECT * FROM Films ORDER BY year, title – несколько полей сортировки

Группы данных

С помощью необязательного предложения GROUP BY создаются группы данных. Это удобно для получения итоговых значений. Например, сколько фильмов выпущено по годам

```
SELECT year, COUNT (*) AS cnt
FROM films GROUP BY year
```

В GROUP BY можно указать столько столбцов, сколько нужно.

В результате группы будут вложенные

Отбор групп

Предложение HAVING используется для отбора групп.

```
SELECT year, COUNT (*) AS cnt
FROM films
GROUP BY year
HAVING COUNT (*) >= 100
```

Итоговые значения

название	описание
COUNT(*)	Возвращает количество строк источника записей
COUNT	Возвращает количество значений в указанном столбце
SUM	Возвращает сумму значений в указанном столбце
AVG	Возвращает среднее значение в указанном столбце
MIN	Возвращает минимальное значение в указанном столбце
MAX	Возвращает максимальное значение в указанном столбце

Соединение таблиц

```
FROM <таблица 1>
  [INNER]
  {{LEFT | RIGHT | FULL } [OUTER]} JOIN <таблица 2>
[ON <предикат>]
```

При этом **INNER JOIN** означает, что в результирующий набор попадут только те соединения строк двух таблиц, для которых значение предиката равно **TRUE**.

Внешнее соединение **LEFT JOIN** означает, что помимо строк, для которых выполняется условие предиката, в результирующий набор попадут все остальные строки из первой таблицы (левой). При этом отсутствующие значения столбцов из правой таблицы будут заменены **NULL**-значениями.
Соединение **RIGHT JOIN** обратно соединению **LEFT JOIN**, то есть в результирующий набор попадут все строки из второй таблицы, которые будут соединяться только с теми строками из первой таблицы, для которых выполняется условие соединения.
Наконец, при полном соединении (**FULL JOIN**) в результирующую таблицу попадут не только те строки, которые имеют одинаковые значения в сопоставляемых столбцах, но и все остальные строки исходных таблиц, не имеющие соответствующих значений в другой таблице. В этих строках все столбцы той таблицы, в которой не было найдено соответствия, заполняются **NULL**-значениями. То есть полное соединение представляет собой комбинацию левого и правого

Подключение из Python

- **Подключение** — объект, в котором указывается либо путь к файлу, либо путь к серверу, также могут быть логин и пароль. Он отвечает только за подключение к БД и, соответственно, отключение от нее
- **Курсор** — объект, в котором непосредственно производится работа с БД

Запросы с параметрами

```
result = cur.execute("""SELECT * FROM films
            WHERE year = ? and duration > ?
""", (2010, 90)).fetchall()

result = cur.execute("""SELECT * FROM Films
            WHERE year = ?""", (2009,)).fet
chall()
```

Добавление

```
INSERT INTO
имя_таблицы(названия_полей)
VALUES(значения)

INSERT INTO genres(id, title)
VALUES(50,'учебные фильмы')

INSERT INTO genres(title)
VALUES('поздравления')

INSERT INTO genres VALUES (45,
'Научные'), (46, 'Сказки')
```

Изменение

```
UPDATE имя_таблицы
SET название_столбца =
новое_значение WHERE условие
```

```
UPDATE films
SET duration = '162'
WHERE title = 'Аватар'
```

Удаление

```
DELETE from имя таблицы
WHERE условие
```

```
DELETE from films
WHERE year < 1920
```

Изменение базы в Python

```
cur = con.cursor()
ids = [20, 30, 31]
cur.execute("DELETE FROM films
        WHERE id IN (" + ", ".join('?' * len(ids))
        + ")", ids)
con.commit()
con.close()
```

Метод commit сохраняет изменения,
rollback – отменяет транзакцию

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru