Курс «Базовая обработка данных на языке Python»

автор: Киреев В.С., к.т.н., доцент

Лабораторная работа № 7

Тема: «Хранение и обработка данных с помощью SQLite»

Цель работы: изучить методы работы в Python с файловыми СУБД, на примере SQLite3.

Теоретическая справка

Модуль OS

Модуль OS в Python предоставляет функции для взаимодействия с операционной системой. OS входит в стандартные служебные модули Python. Этот модуль предоставляет переносимый способ использования функциональности, зависящей от операционной системы. Модули *os* и *os.path* включают множество функций для взаимодействия с файловой системой.

Для получения местоположения текущего рабочего каталога используется os.getcwd().

```
import os
cwd = os.getcwd()
```

С помощью метода os.mkdir() в Python создается каталог с именем path с указанным числовым режимом. Этот метод вызывает FileExistsError, если создаваемый каталог уже существует.

```
import os
directory = "CurDir"
parent_dir = "ParDir"
path = os.path.join(parent_dir, directory)
os.mkdir(path)
```

Метод os.remove() в Python используется для удаления или стирания пути к файлу. Этот метод не может удалить или стирать каталог. Если указанный путь является каталогом, то метод вызовет ошибку OSError.

В функции os.path.getsize( ) python даст размер файла в байтах. Для использования этого метода нужно передать имя файла в качестве параметра.

```
import os
size = os.path.getsize("filename")
```

SQLite

SQLite — это легкий, бессерверный, автономный и высоконадежный движок базы данных SQL . Он широко используется благодаря своей простоте, легкости настройки и нулевой конфигурации. SQLite соответствует требованиям ACID и реализует большинство

стандартов SQL, используя динамически и слабо типизированный синтаксис SQL, который не гарантирует целостность домена.

SQLite оперирует концепцией классов хранения или storage class. И по сути все эти пять типов называются классами хранения. Концепция классов хранения несколько шире, чем тип данных. Например, класс INTEGER по сути объединяет 6 различных целочисленных типов данных разной длины.

Тип	Описание	Соответствие другим типам
NULL	Значение равно NULL.	
INTEGER	Значение представляет собой целое число со знаком, хранящееся в 1, 2, 3, 4, 6 или 8 байтах в зависимости от величины значения.	INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8
REAL	Значение представляет собой значение с плавающей запятой, хранящееся как 8-байтовое число с плавающей запятой IEEE.	REAL, DOUBLE, DOUBLE PRECISION, FLOAT
TEXT	Значение представляет собой текстовую строку, сохраненную с использованием кодировки базы данных (UTF-8, UTF-16BE или UTF-16LE).	CHARACTER(20), VARCHAR(255), VARYING CHARACTER(255), NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100), TEXT, CLOB
BLOB	Значение представляет собой блок данных, хранящийся точно в том виде, в котором он был введен.	

SELECT QUERY используется для извлечения данных из базы данных.

```
SELECT column_name AS alias_name

FROM table_name
```

ALTER TABLE в SQLite используется для изменения структуры таблицы, которая уже присутствует в схеме базы данных. SQLite поддерживает только некоторые функции команды ALTER TABLE, чего нет в других базах данных SQL

```
ALTER TABLE  table_name  ADD COLUMN  new_column  column_definition;
```

Вставка данных

```
insert into students(stu_id, first_name, fees, email)

VALUES

(1, 'Amulya', 50000, "ammuamulya@gmail"),

(2, 'Ram', 30000, "ramyerra@gmail");
```

В SQLite **соединения** бывают разных типов. Некоторые из них определены ниже:

- внутреннее соединение
- левое внешнее соединение (левое соединение)
- перекрестное соединение

Но, однако, RIGHT OUTER JOIN и FULL OUTER JOIN не поддерживаются в SQLite.

```
SELECT t.name, t.salary, d.dept
FROM Teachers as t
LEFT OUTER JOIN department as d
on t.Id = d.emp_id;
```

Агрегатные функции

Агрегатная функция — это функция, которая выполняет вычисления над набором значений и возвращает одно значение. Агрегатные функции часто используются с GROUP BY предложением оператора SELECT . GROUP BY. Предложение разбивает результирующий набор на группы значений, а агрегатная функция может использоваться для возврата одного значения для каждой группы. Наиболее часто используемые агрегатные функции — AVG, COUNT, DENSE_RANK, MAX , MIN , RANK , SUM .

```
SELECT MIN(Price)
FROM Products;
```

Транзакции позволяют сгруппировать несколько инструкций SQL в одну единицу работы, которая фиксируется в базе данных как одна атомарная единица. При сбое любой инструкции в транзакции можно выполнить откат изменений, выполненных предыдущими инструкциями. Начальное состояние базы данных сохраняется при запуске транзакции. Использование транзакций может также повысить производительность SQLite при одновременном внесении многочисленных изменений в базу данных.

В SQLite только одна транзакция может иметь изменения, ожидающие внесения в базе данных, в любой конкретный момент времени. В связи с этим время ожидания вызовов BeginTransaction и методов Execute в SqliteCommand может истечь, если другая транзакция занимает слишком много времени.

Транзакции сериализуемы по умолчанию в SQLite. Этот уровень изоляции гарантирует, что любые изменения, внесенные в транзакцию, будут полностью изолированы. Изменения транзакции не затрагивают другие инструкции, выполняемые за пределами транзакции.

SQLite также поддерживает чтение незафиксированных изменений при использовании общего кэша. Этот уровень допускает "грязные" и неповторяемые операции чтения, а также фантомы:

- "Грязное" чтение происходит, когда изменения, ожидающие в одной транзакции, возвращаются запросом вне этой транзакции, но при этом изменения в транзакции откатываются. Результаты содержат данные, которые никогда не фиксировались в базе данных.
- Неизменяемое чтение происходит, когда транзакция запрашивает одну и ту же строку дважды, но результаты отличаются, так как они были изменены между двумя запросами другой транзакцией.
- Фантомы — это строки, которые изменяются или добавляются для соответствия предложению запроса where во время транзакции. Если они разрешены, один и тот же запрос может возвращать разные строки при двукратном выполнении в одной транзакции.

Оконные функции — мощный инструмент языка SQL, позволяющий проводить сложные вычисления по группам строк, которые связаны с текущей строкой. В стандартном SQL-запросе все наборы строк рассматриваются как один сплошной блок данных, для которого и вычисляются агрегатные значения. Однако, когда применяются оконные функции, запрос сегментируется на группы строк (или «окна»), и для каждого такого сегмента подсчитываются индивидуальные агрегатные значения. Это окно, которое подаётся в оконную функцию, может быть:

- всей таблицей
- отдельными партициями таблицы, то есть группой строк на основе одного или нескольких полей
- конкретным диапазоном строк в пределах таблицы или партиции.

```
SELECT column_name1,
 window_function(column_name2)
 OVER([PARTITION BY column_name1] [ORDER BY column_name3]) AS new_column
FROM table_name;
```

Оконная функция row_number() присваивает последовательные целые числа каждому ряду в порядке выполнения условия «ORDER BY» в window-defn (в данном случае «ORDER BY y»). Обратите внимание, что это не влияет на порядок, в котором возвращаются результаты из общего запроса.

Порядок вывода результатов по-прежнему определяется предложением ORDER BY, присоединенным к оператору SELECT (в данном случае «ORDER BY x»).Именованные предложения window-defn также могут быть добавлены в оператор SELECT с помощью предложения WINDOW и затем упоминаться по имени в вызовах оконных функций.

```
SELECT x, y, row_number() OVER win1, rank() OVER win2

FROM t0

WINDOW win1 AS (ORDER BY y RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT
ROW),

       win2 AS (PARTITION BY y ORDER BY x)

ORDER BY x;
```

Самостоятельное задание

1. Скачать, используя api.hh.ru, 500 вакансий, содержащих в названии "Разработчик", с указанием ЗП

   1.1. Оставить только id, name, area.name, salary.from, salary.to, salary.gross, salary.currency, employer.name

   1.2. Используя средства pandas, сохранить эти вакансии в виде df.csv, df.json.

2. Создать пустые базы db1 и db2 в Sqlite3, с полями из 1.1
3. Наполнить базы, используя, соответственно считывание из csv и из json.
4. Создать пустую базу db3. Создать три таблицы - вакансии, города, и работодатели. Нормализовать таблицу из 1.1, и разбить ее данные на 3 перечисленные таблицы.
5. Средствами SQL создать новую колонку очищенной зарплаты в db3, содержащую
6. среднюю зп по полям from и to, пересчитанную в рубли и очищенную от налога (для простоты - 13%)
7. По таблицам db3 с помощью JOIN, GROUP BY, HAVING, ORDER BY,
8. вывести по убыванию очищенной зарплаты список вакансий в городах Москва и Ташкент
9. С помощью оконных функций (dense_rank) вывести в каждом городе вакансию с максимальной очищенной зп
10. С помощью оконных функций (lag, lead, next) для каждой вакансии вывести макс и мин очищенную зп