Python

15

Базы данных

База данных(Data base, DB) - набор сведений, хранящихся неким упорядоченным способом.

База данных - контейнер(обычно файл или группа файлов) для хранения упорядоченных данных

Таблицы

Таблицы - структурированный файл, в котором могут храниться данные определенного типа.

Таблица - Структурированный набор данных определенного типа.

Схема БД

Схема - информация о том, какие данные могут храниться в таблицах, как они распределены по таблицам, как называются отдельные информационные блоки и т.д.

Схема - информация о базе данных, а также о структуре и свойствах ее таблиц.

Столбцы

Таблицы состоят из столбцов, в которых хранятся отдельные фрагменты информации

Столбец - одиночное поле таблицы

Тип данных

Типы данных ограничивают характер информации, которую можно хранить в столбце. Типы данных также помогают корректно отсортировать информацию.

http://unetway.com/tutorial/sql-tipy-dannyh/

Строки

Данные в таблице хранятся в строках, и каждая запись хранится в своей строке.

Строка - отдельная запись в таблице

Первичные ключи

Первичные ключи - столбец(или набор столбцов), значения которого уникально идентифицируют каждую строку таблицы.

SQL

SQL (Structured Query Language) - язык структурированных запросов, который был специально разработан для взаимодействия с базами данных. Задача SQL - предоставить простой и эффективный способ чтения и записи информации из баз данных

СУБД

СУБД - система управления базами данных

СУБД: Oracle, MySQL, PostgreSQL, MongoDB….

Pycharm создание базы sqlite

View -> Tool Buttons -> Database -> + -> Data source -> Sqlite -> + -> OK

Создание таблицы

```
CREATE TABLE person (
    id integer,
    firstname varchar,
    lastname varchar
)
```

Комментарии

-- комментарий

/*

Многострочный комментарий

*/

Удаление таблицы

```
DROP TABLE person;
```

Изменение таблицы

```
ALTER TABLE person

ADD COLUMN email varchar(255);


ALTER TABLE person

DROP COLUMN email; -- не работает для sqlite
```

Добавление данных

```
INSERT INTO person (id, firstname, lastname)

VALUES (0, 'Alex', 'Varkalov');
```

Выборка данных

```
SELECT * FROM person;

SELECT id FROM person;

SELECT firstname, lastname FROM person;
```

Выборка данных с условием

```
SELECT * FROM person WHERE firstname = 'Alex';
```

Обновление данных

```
UPDATE person

SET firstname = 'Alexander'

WHERE firstname = 'Alex';
```

Удаление данных

```
DELETE FROM person

WHERE id = 1;
```

AND, OR

```
SELECT * FROM person

WHERE firstname = 'Alex' and lastname = 'Varkalov';

SELECT * FROM person

WHERE firstname = 'Alex' or id < 2;
```

Задание

```
CREATE TABLE user (

   id integer primary key autoincrement,

   firstname varchar(255),

   lastname varchar(255),

   age integer

)
```

1. Создать таблицу
2. Добавить 5 записей
3. Получить всех пользователей с вашим именем
4. Получить всех пользователей младше 25
5. Получить всех пользователей в возрасте от 15 до 18

Sqlalchemy

sqlalchemy - библиотека для работы с базами данных

pip install sqlalchemy

Выполнение запросов к базе с sqlalchemy

```
from sqlalchemy import create_engine

e = create_engine("sqlite:///test.db")
e.execute("""
  create table user (
     id integer primary key autoincrement,
     firstname varchar,
     lastname varchar
  )
""")
```

create_engine - создание подключения к базе.

.execute - выполнение запроса к базе

Добавление записей

```
from sqlalchemy import create_engine

e = create_engine("sqlite:///test.db")
e.execute("""
   insert into user (firstname, lastname)
      values ('Alex', 'Varkalov')
""")
```

Получение данных

```
from sqlalchemy import create_engine

e = create_engine("sqlite:///test.db")
result = e.execute("""select * from user""")
for user in result:
    print(user)
```

Транзакция

```
from sqlalchemy import create_engine

engine = create_engine("sqlite:///test.db")
conn = engine.connect()
trans = conn.begin()
conn.execute(
   "insert into user (firstname, lastname)
values (:firstname, :lastname)",
   firstname="Pasha", lastname='Ivanov')
trans.commit()
conn.close()
```

Транзакция — это осуществление одного или нескольких изменений базы данных.

.commit - применение транзакции
.rollback - отмена транзакции

ORM - Object related model

```
from sqlalchemy import create_engine
from sqlalchemy import Table, Column, Integer, String, MetaData
from sqlalchemy.orm import mapper

engine = create_engine("sqlite:///test1.db", echo=True)
metadata = MetaData()
users_table = Table('user', metadata,
   Column('id', Integer, primary_key=True),
   Column('firstname', String),
   Column('lastname', String),
)
metadata.create_all(engine)
class User:
   def __init__(self, firstname, lastname):
      self.firstname = firstname
      self.lastname = lastname
mapper(User, users_table)
user = User('Alex', 'Varkalov')
```

ORM - Object related model

```
from sqlalchemy import create_engine
from sqlalchemy import Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base

engine = create_engine("sqlite:///test1.db", echo=True)
Base = declarative_base()
class Person(Base):
   __tablename__ = 'person'
   id = Column(Integer, primary_key=True)
   firstname = Column(String)
   lastname = Column(String)

   def __init__(self, firstname, lastname):
      self.firstname = firstname
      self.lastname = lastname
Base.metadata.create_all(engine)
```

Session

```
from sqlalchemy.orm import sessionmaker
Session = sessionmaker(bind=engine)
session = Session()
session.add(Person('Alex', 'Varkalov'))
session.add_all([Person('Alex', 'Petrov'), Person('Ann', 'Ivanova')])
session.commit()
```

Создание запроса Query

```
person = session.query(Person).filter_by(
   firstname="Alex").first()

person = session.query(Person).filter(
   Person.firstname=="Alex").first()

person = session.query(Person).filter(and_(
  Person.firstname=="Alex",
  Person.lastname=="Varkalov",
)).all()
```