后端3

python基础

https://github.com/jackfrued/Python-100-Days

https://www.datacamp.com/tutorial/introduction-fastapi-tutorial

https://www.sqlitetutorial.net/sqlite-sample-database/

测试数据

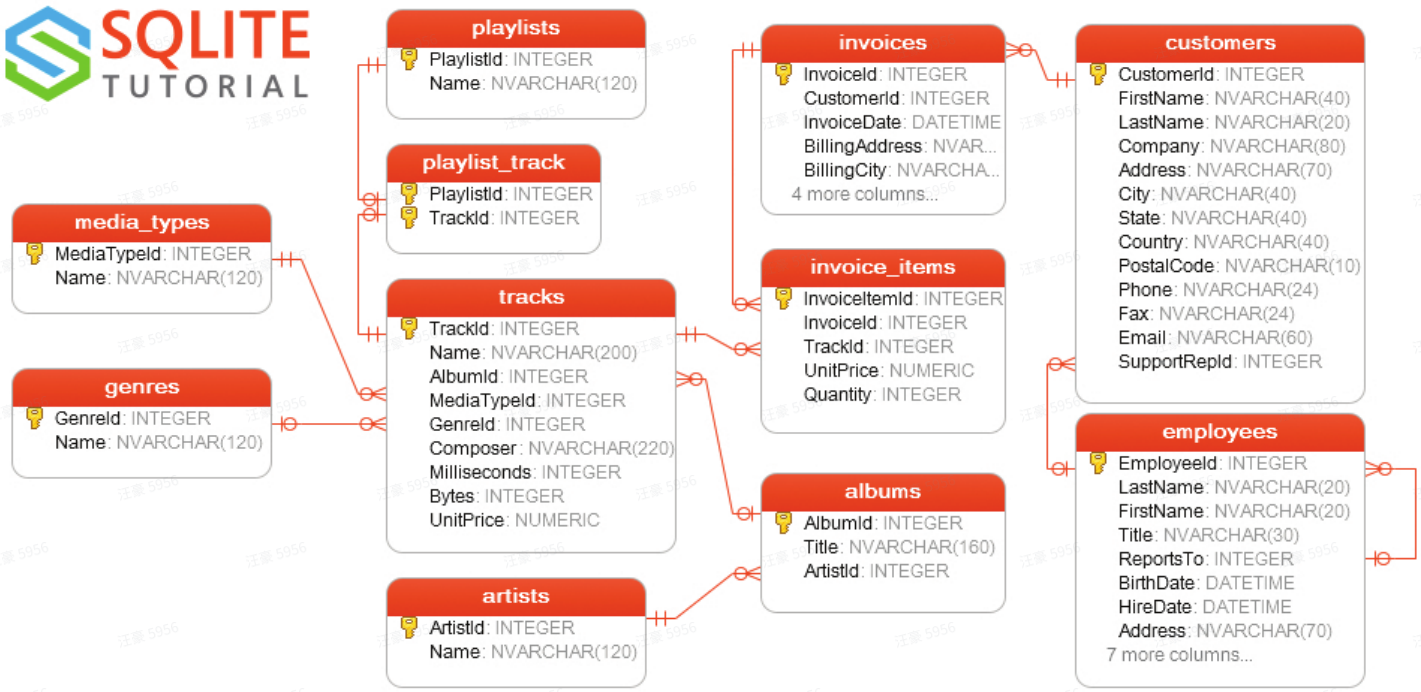
Chinook

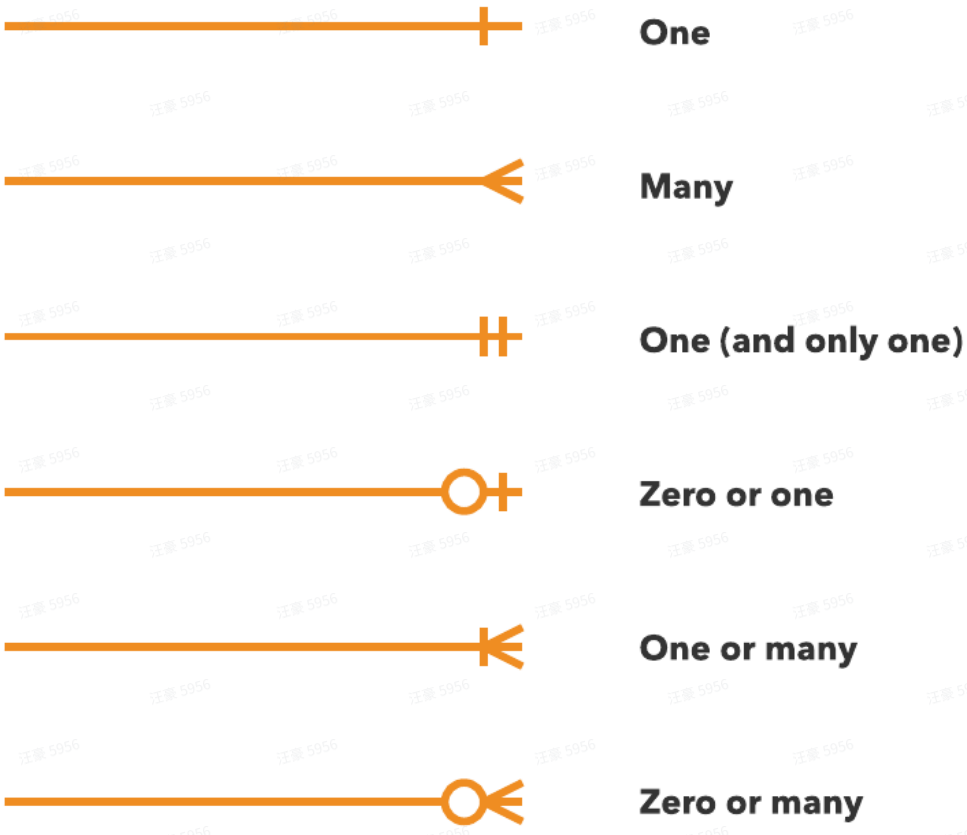
https://github.com/lerocha/chinook-database/releases

Chinook示例数据库包含11个表，如下：

1. `employees` (员工表)存储员工数据，例如 ID、姓氏、名字等。它还有一个名为 ReportsTo 的字段来指定谁向谁报告。
2. `customers` (客户表)存储客户数据。
3. `invoices` (发票抬头表)存储发票抬头数据。与发票明细表是一对多关系。
4. `invoice_items` (发票明细表)存储发票明细数据。
5. `artists` (艺术家表)存储艺术家数据。这是一个包含 ID 和姓名的简单表。
6. `albums` (专辑表)存储了曲目列表的数据。每张专辑属于一位艺术家，但一位艺术家可能有多张专辑。
7. `media_types` (媒体类型表)存储媒体类型，例如 MPEG 音频和 AAC 音频文件。
8. `genres` (流派表)存储摇滚、爵士、金属等音乐类型。
9. `tracks` (曲目表)存储歌曲的数据。每首曲目属于一张专辑。
10. `playlists` (播放列表)存储播放列表的数据。每个播放列表包含一个曲目列表。每个曲目可能属于多个播放列表。
11. `playlist_track` (播放列表与曲目中间表)播放列表表和曲目表之间的关系是多对多的。此表用于反映这种关系。

E-R图


SQLITE TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...


One
Many
One (and only one)
Zero or one
One or many
Zero or many

```
# 使用pip安装 postgresql 客户端
pip install pgcli
# 连接pg
pgcli.exe  -U u1 -h 41e1af074877.c.methodot.com -p 30290 -d chinook
# 密码是1234qwer
```

使用 pgcli 命令行连接数据库:

代码块

```
PS C:\Users\RYefccd\workspace\fastapidemo>
C:\users\ryefccd\appdata\local\packages\pythonsoftwarefoundation.python.3.12_qb
z5n2kfra8p0\localcache\local-packages\python312\Scripts\pgcli.exe  -U u1 -h
41e1af074877.c.methodot.com -p 30290 -d chinook
Password for u1:
```

进入pgcli命令行后可以使用 \d 查看当前数据库的所有表(等价于mysql 的 show tables)

代码块

```

Using local time zone Asia/Shanghai (server uses UTC)
Use `set time zone <TZ>` to override, or set `use_local_timezone = False` in
the config
Server: PostgreSQL 10.5 (Debian 10.5-1.pgdg90+1)
Version: 4.3.0
Home: http://pgcli.com
u1@41e1af074877:chinook> \d
+--------+----------------+-------+-------+
| Schema | Name           | Type  | Owner |
|--------+----------------+-------+-------/
| public | album          | table | u1    |
| public | artist         | table | u1    |
| public | customer       | table | u1    |
| public | employee       | table | u1    |
| public | genre          | table | u1    |
| public | invoice        | table | u1    |
| public | invoice_line   | table | u1    |
| public | media_type     | table | u1    |
| public | playlist       | table | u1    |
| public | playlist_track | table | u1    |
| public | track          | table | u1    |
+--------+----------------+-------+-------+
SELECT 11
```

```
24  Time: 0.054s
25  u1@41e1af074877:chinook>
```

一对多查询

查询顾客在哪个员工上买过专辑.

代码块

```
1  select
   customer.first_name,customer.last_name,customer.email,employee.first_name,emplo
   yee.last_name from customer join emp
2   loyee ON employee.employee_id = customer.support_rep_id;
```

多对多查询

查询歌单和歌曲的映射关系.

代码块

```
1  select playlist.playlist_id, track.track_id,playlist.name as "playname",
   track.name as "musicname" from track join playlis
2   t_track ON playlist_track.track_id = track.track_id join playlist ON
   playlist.playlist_id = playlist_track.playlist_id;
```

聚合分析

查询不同歌单中有多少歌曲.

代码块

```
1  with tmp as (select playlist.playlist_id, track.track_id,playlist.name as
   "playname", track.name as "musicname" from track
2    join playlist_track ON playlist_track.track_id = track.track_id join
   playlist ON playlist.playlist_id = playlist_track.playlist_id)
3   select playname,count(*) from tmp group by playname;
```

更多数据分析练习: Chinook数据分析

fastapi

https://fastapi.org.cn/

安装fastapi及其依赖

代码块

```
pip install fastapi==0.115.12
pip installpsycopg==3.2.9
pip install uvicorn==0.34.3
```

运行程序

假设是main.py和pgdemo.py程序:

代码块

```
# 运行main.py
uvicorn.exe  main:app
# 运行 pgdemo.py
uvicorn.exe  pgdemo:app
```

main.py

代码块

```
from typing import Union
from fastapi import FastAPI, Request
app = FastAPI()

@app.get("/", tags=["test"])
@app.put("/", tags=["test"])
@app.post("/", tags=["test"])
def mytest(request: Request):
    return {"Hello": "World", "http_method": request.method}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: Union[str, None] = None):
    return {"item_id": item_id, "q": q}

```

pgdemo.py

代码块

```python
from typing import Union
from fastapi import FastAPI, Request
from pydantic import BaseModel
import psycopg

# https://www.postgresql.org/docs/current/libpq-connect.html#LIBPQ-CONNSTRING
# https://www.psycopg.org/psycopg3/docs/api/conninfo.html
# https://www.psycopg.org/psycopg3/docs/basic/usage.html
# psql -U u1 -h 41e1af074877.c.methodot.com -p 30290 -d chinook 1234qwer
# conn = psycopg.connect("dbname=test user=postgres")
conn = psycopg.connect(
                    host="41e1af074877.c.methodot.com",
                    port=30290,
                    dbname="chinook",
                    user="u1",
                    password="1234qwer")

app = FastAPI()

class Customer(BaseModel):
    # 定义客户模型用于插入数据和修改数据.
    customer_id: int
    first_name: str
    last_name: str
    company: str
    address: str
    city: str
    state: str
    country: str
    postal_code: str
    phone: str
    fax:str
    email: str

@app.get("/customer/{item_id}", tags=["customer"])
def get_customer(item_id: int):
    cur = conn.cursor()
    cur.execute("select * from customer where customer_id=%s;", [item_id])
    res = cur.fetchone()
    cur.close()
    return {"Hello": "World", "customer": res}
```

```python

@app.post("/customer", tags=["customer"])
def post_customer(customer: Customer):
    cur = conn.cursor()
    ## 下次讲 数据库的插入和修改. 大家可以提前预习.
    cur.close()
    return {"Hello": "World"}

```