Cruise Earth，
一款可视化环球船只信息数据中心

数据库原理课程设计报告

学　　号　2150233

姓　　名　朱申哲

专　　业　基础学科拔尖基地（计算机）

授课老师　李文根

目录

摘 要

“Cruises Earth”是一个全球船只信息管理和展示的数据中心系统，旨在为全球航运业提供高效、直观和安全的船只管理平台。随着全球贸易和旅游业的发展，海运作为国际物流和旅游的重要组成部分，显现出其不可或缺的地位。然而，船只管理和航行数据的复杂性和分散性，给航运公司、港口管理部门以及相关利益者带来了巨大的挑战。因此，开发一个集成化的船只信息管理系统显得尤为重要。“Cruises Earth”系统整合了谷歌地图服务，提供了普通模式（Normal）、街景模式（Street）和遥感模式（Satellite）三种地图显示模式。这一功能不仅满足了用户对不同视角下船只位置信息的需求，还增强了数据的可视化效果，使用户能够更直观地理解和操作船只数据。通过这一系统，用户可以实时查看船只的当前位置、轨迹记录，并进行路径规划，极大地提升了船只管理的效率和精确度。系统的核心功能包括船只信息管理、轨迹管理和地图显示。船只信息管理模块允许用户注册、查询和修改船只的基本信息，确保数据的实时性和准确性。轨迹管理模块通过记录和存储船只的航行路径，为用户提供详细的航行历史数据和轨迹回放功能，帮助用户分析和优化航行路线。此外，系统集成的谷歌地图服务支持三种地图显示模式，用户可以根据需要自由切换，获得最佳的视觉体验和信息获取。系统设计采用高可用架构，包括主从复制、负载均衡和自动故障转移，确保系统24/7不间断运行。系统支持水平扩展，能够根据需求动态调整资源，处理不断增长的船只数量和用户请求。综上所述，“Cruises Earth”通过整合先进的地图服务和全面的船只信息管理功能，提供了一个高效、直观和安全的船只管理平台。该系统不仅能满足全球用户的需求，还能确保数据的安全性和合规性，为全球航运业提供有力支持。通过“Cruises Earth”，用户能够更加便捷地管理和分析船只数据，从而提升运营效率，优化航行路线，降低运营成本，为航运业的数字化转型贡献力量。

关键词：数据库；船只管理；可视化；在线地图；海洋

一.概述

1.1 课题背景

“Cruises Earth”是一个环球船只信息数据中心，用类似谷歌地图的方式 2D 展示，支持多种显示模式。系统会记录每一艘船只过去一个月的轨迹，并且实时展示海域的天气状况。数据库记录的信息主要有：船只轨迹、海港信息、每艘船只的信息（包括吨位、所属公司、详细链接等）。

随着全球贸易的不断发展，航运业的规模和复杂度也在不断增加。船只数量的增加、航线的增多、船舶技术的不断更新等因素，使得船只信息的管理变得越来越重要。“Cruises Earth”为船只信息的管理提供了一种全新的方式，可以更好地满足航运业的需求。

综上，“Cruises Earth”是一个旨在收集、存储、分析和共享全球船只信息的系统。该系统可以帮助船运公司、港口管理机构、海事监管机构等各方更好地了解全球船只的运行情况，提高航运安全性，优化航线规划，降低运营成本，促进航运业的可持续发展。

1.2 编写目的

“Cruises Earth”项目作为一个全球船只信息管理和展示的数据中心系统，具有广泛而深远的意义。该项目的重要性及其对各相关领域的影响有以下几点。

- 提升航运管理效率

全球航运业面临着大量数据的管理需求，包括船只的位置信息、航行轨迹、维护记录、货物信息等。传统的手工管理方式效率低下且易出错。“Cruises Earth”通过提供一个集中化、系统化的数据管理平台，使得航运公司和港口管理部门能够高效管理船只信息，减少人工操作的失误，提高整体运营效率。

- 增强数据的可视化和透明度

“Cruises Earth”系统整合了谷歌地图服务，提供普通模式、街景模式和遥感模式三种

地图显示选项。用户可以直观地查看船只的当前位置和航行轨迹。这种可视化的展示方式不仅使得数据更易于理解和分析，还增强了信息的透明度，有助于利益相关者实时了解船只的状况和位置，提高决策的准确性和及时性。

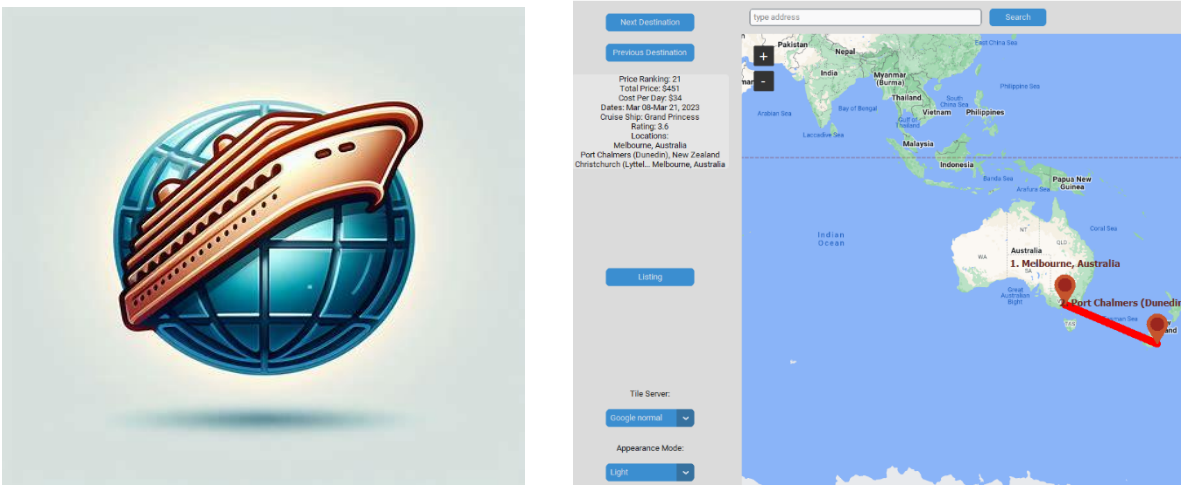
- 促进全球航运业的协同和合作

“Cruises Earth”系统的全球化设计支持多语言界面和多种区域设置，能够满足不同国家和地区用户的需求。通过提供一个统一的平台，不同地区的航运公司、港口管理部门和其他相关机构可以更好地协同工作，分享信息，优化航运路线和运营计划，促进全球航运业的合作与发展。

- 推动航运业的数字化转型

随着信息技术的发展，传统行业的数字化转型已成为必然趋势。“Cruises Earth”项目通过引入先进的数据管理和地图服务技术，为航运业的数字化转型提供了有力支持。通过该系统，航运业可以更好地利用数据驱动业务决策，提高运营效率，降低成本，增强市场竞争力。

总的来说，“Cruises Earth”项目旨在通过整合先进的地图服务和全面的船只信息管理功能，为用户提供高效、直观和安全的船只管理平台。通过实施上述功能和安全措施，系统不仅能满足全球用户的需求，还能确保数据的安全性和合规性，从而为全球航运业提供有力支持。



左图为概念图，右图为地图界面图

二.需求分析

2.1 功能需求

对于不同的用户（使用者和管理者）而言，数据库的整体功能需求如下：

User TypeRequirement

船只：

I 类（国际）船只：远洋货轮，油轮，邮轮等 定时上报当前位置、状态（在港/出港），记录轨迹，状态定时更新（位置信息，空闲/载客，工作状态等）

到达港口上报情况

II 类（国内）船只： 一般货船，游轮等

III 类（小型）船只：小型货船，渔船等

港口 定时更新（港口船只情况，天气状况，以及是否出现突发状况）

船主：

I 类（国际）船只船主 获取目标港口、全球海域情况、天气、重要航线、全球船只的信息

II 类（国内）船只船主获取目标港口、海域水域情况、周边船只的信息

III 类（小型）船只船主 获取目标港口、海域水域情况、周边船只的信息

港口管理员：

统计船只情况、入港出港时长等信息，优化

系统管理员： 添加/删除/更改船只信息

一般用户： 查询各个船只、港口情况，无上传权限

详细功能需求如下：

I 类（国际）船只：

主要由远洋货轮，油轮，邮轮这些大型船只组成。这些大型船只的价值较高，目的地和出发地均为国际著名港口。这类船只相对而言比较重要，因此需要不断上传船只的位置信息和运行状况。同时，此类船只的出发点和目的地以及航线基本已经确定，载客和装货情况也是在

发车前确定，在航行过程中只需要定时地上报位置信息和当前交通工具的状态即可。

II 类（国内）船只：

主要由一般货船，游轮等。提供实时的船只位置和航行状态，这对于这类船只来说非常重要，因为近海的水域比大洋船只更密集，它们需要知道自己的位置和航行状态，以便更好地规划航线和避免危险。船只定时上传历史轨迹，因为它们需要了解自己的历史轨迹，以便更好地规划航线和避免危险。

III 类（小型）船只：

同样，这类船只需要上传实时的船只位置和航行状态。但对于这类船只，由于密度较大，还需要提供船只的详细信息和规格，例如船只的长度、宽度、吨位、载重等。这类船只也需要定时上传历史轨迹，因为它们需要了解自己的历史轨迹，以便更好地规划航线和避免危险。同时，这类船只需要保证自己在可活动的范围内。

港口：

港口需要对船只的进出港报告进行管理，以便更好地掌握船只的位置和航行状态，确保船只的安全和监控。因此，港口需要掌握船只的信息进行管理，例如船只的长度、宽度、吨位、载重等，以便更好地规划航线和运输货物。同时，港口需要将船只进出港报告管理实时上传至数据库。

I 类（国际）船只船主：

I 类（国际）船只船主目标是一个确定的航班或车次，不会进行临时性的路线重规划，其主要需求是接受所驾驶的船只的实时状态信息，是否发生晚点、延误等情况，从而安排船只路线。

II 类（国内）船主：

II 类以国内中小型船主为主，其对线路和车次没有相应管理权限，其主要需求是了解周边船只实时位置信息和港口信息，通过该信息调整相应路线以避免相邻船只之间相距过近等情况产生。

III 类（小型）船只船主：

III 类船只船主以小型渔船，游艇为主。这类船的特点是航程短，体型小；因此这类船只的主要诉求是能够及时上传位置信息以及接受周围的求助信号。同时，系统必须提醒这类船只在航程允许的范围内航行，以免出现事故。

系统管理员：

系统管理员包括由电脑进行自动化控制的管理者和生活意义上的系统管理者，其主要需求是添加、删除或者修改船只信息。同时，系统管理者还需对现有的船只和港口信息进行维护和检测。

港口管理员：

港口管理员的主要需求是获取港口所有船只的时空承载信息，最好以可视化的方式进行呈现，对其进行分析并进行港口的安排或者是其他的调整。同时，港口管理员还可以从数据当中获得船只进出对港口效率比的影响，从而完善地评估港口发展和未来的规划。

一般用户：

一般用户被认为是一般的船只爱好者。他们不具备任何上传信息的权利，只有查看的权利。查看的范围仅限于 I 类船只和港口相关情况。这是考虑到 II 类和 III 类的隐私信息。

在系统层面，

我认为，“Cruise Earth”这种系统需要具备以下特点：

全球性：该系统应该能够覆盖全球范围内的船只，除了敏感船只。

实时性：该系统应该能够实时监测船只的位置、状态、速度等信息，以便及时做出调整。

安全性：该系统应该能够保证船只的安全，防止船只遭受海盗袭击、意外碰撞等危险。

易用性：该系统应该易于使用，用户可以方便地查看船只的信息、位置等。

可靠性：该系统应该能够保证数据的准确性和可靠性，以便用户做出正确的决策。

作为一个实时的全球船只平台，设计过程中必须要考虑到公交系统的时空波动性，这种波动性主要表现为高峰时期、热点所产生的突发的、巨大的访问量，系统管理员要能够可视化地监控系统中的负载和波动，并且能够实现实时的维护。当系统发生故障时，系统应当有备份机制从而使得系统快速地从之前发生的故障中恢复并保证数据不丢失。

2.2 数据字典

船只管理数据库数据字典如下：

Ship

列名	数据类型	约束	描述
ShipID	INT	PRIMARY KEY	船只唯一标识符
Name	VARCHAR(100)	NOT NULL	船只名称
BuildYear	YEAR		建造年份
Capacity	INT		容纳乘客数量
Flag	VARCHAR(50)		船只国旗

Cruise

列名	数据类型	约束	描述
CruiseID	INT	PRIMARY KEY	邮轮唯一标识符
ShipID	INT	FOREIGN KEY	所属船只的标识符
StartDate	DATE	NOT NULL	邮轮开始日期
EndDate	DATE	NOT NULL	邮轮结束日期
Route	VARCHAR(255)		邮轮路线

Visit

列名	数据类型	约束	描述
VisitID	INT	PRIMARY KEY	访问唯一标识符
CruiseID	INT	FOREIGN KEY	邮轮标识符
PortID	INT	FOREIGN KEY	港口标识符
ArrivalDate	DATETIME	NOT NULL	到达日期和时间

DepartureDate | DATETIME | NOT NULL | 离开日期和时间

Port

列名	数据类型	约束	描述
PortID	INT	PRIMARY KEY	港口唯一标识符
Name	VARCHAR(100)	NOT NULL	港口名称
Country	VARCHAR(50)	NOT NULL	所在国家
City	VARCHAR(50)		所在城市

Cabin

列名	数据类型	约束	描述
CabinID	INT	PRIMARY KEY	舱房唯一标识符
ShipID	INT	FOREIGN KEY	船只标识符
CabinNumber	VARCHAR(10)	NOT NULL	舱房编号
Deck	INT		所在甲板层数
Capacity	INT		可容纳乘客数量

Passenger

列名	数据类型	约束	描述
PassengerID	INT	PRIMARY KEY	乘客唯一标识符
FirstName	VARCHAR(50)	NOT NULL	乘客名
LastName	VARCHAR(50)	NOT NULL	乘客姓
DateOfBirth	DATE		出生日期
Gender	VARCHAR(10)		性别
PassportNumber	VARCHAR(20)	UNIQUE	护照编号

Tour

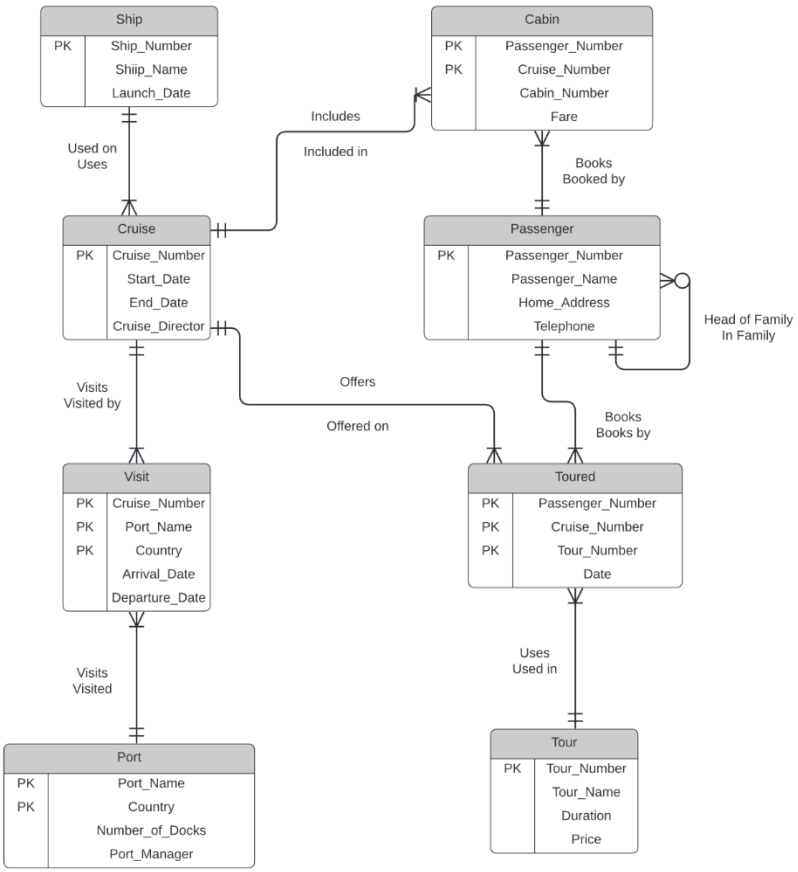
列名	数据类型	约束	描述
TourID	INT	PRIMARY KEY	游览唯一标识符
VisitID	INT	FOREIGN KEY	访问标识符
PassengerID	INT	FOREIGN KEY	乘客标识符
TourName	VARCHAR(100)	NOT NULL	游览名称
TourDate	DATE	NOT NULL	游览日期
Price	DECIMAL(10,2)		游览价格

外键约束如下：

1. Cruise 表中的 ShipID 参考 Ship 表中的 ShipID。
2. Visit 表中的 CruiseID 参考 Cruise 表中的 CruiseID。
3. Visit 表中的 PortID 参考 Port 表中的 PortID。
4. Cabin 表中的 ShipID 参考 Ship 表中的 ShipID。

5. Tour 表中的 VisitID 参考 Visit 表中的 VisitID。
6. Tour 表中的 PassengerID 参考 Passenger 表中的 PassengerID。

2.3 数据流图



三.可行性分析

3.1 技术可行性

经过充分论证，我认为该项目在技术上是完全可行的。主要有以下几点。

1、 系统架构和技术实现

“Cruises Earth”系统采用现代化的技术架构，整合了谷歌地图服务，并利用云计算、

大数据、物联网（IoT）和卫星遥感（RS）等先进技术，实现对船只信息的全面管理和实时展示。云计算平台能够提供高扩展性和高可用性，支持大规模数据处理和存储，确保系统在高并发访问情况下的稳定运行。

2、 数据集成和实时更新

系统能够通过卫星和海上传感器网络实时获取船只位置和状态数据，并进行实时更新和处理。利用物联网技术，可以实现对船只传感器数据的采集和传输，确保数据的实时性和准确性。此外，系统的数据加密和多层次安全措施保证了数据在传输和存储过程中的安全性。

3、 用户界面和体验

系统提供友好和直观的用户界面，用户可以通过普通模式、街景模式和遥感模式自由切换，直观地查看船只信息和轨迹。利用 AI 技术，系统能够提供智能路径规划和预测分析功能，提升用户体验和操作效率。

3.2 应用可行性

随着全球贸易和旅游业的快速发展，海运需求持续增长。全球航运市场对高效、安全、实时的船只管理系统有着强烈的需求。特别是在疫情背景下，航运业需要更加精准和灵活的管理工具来应对不确定性和市场变化。

与传统的船只管理系统相比，“Cruises Earth”具有明显的技术和功能优势。其集成的地图服务和实时数据更新功能，可以提供更全面和精确的船只管理解决方案，满足用户多样化和个性化的需求，具有较强的市场竞争力。

项目实施将分为多个阶段，包括需求分析、系统设计、开发测试、部署实施和维护升级。每个阶段都有明确的目标和任务，确保项目按计划顺利进行。通过项目管理和质量控制，确保系统的高质量和高可靠性。

四.概念设计

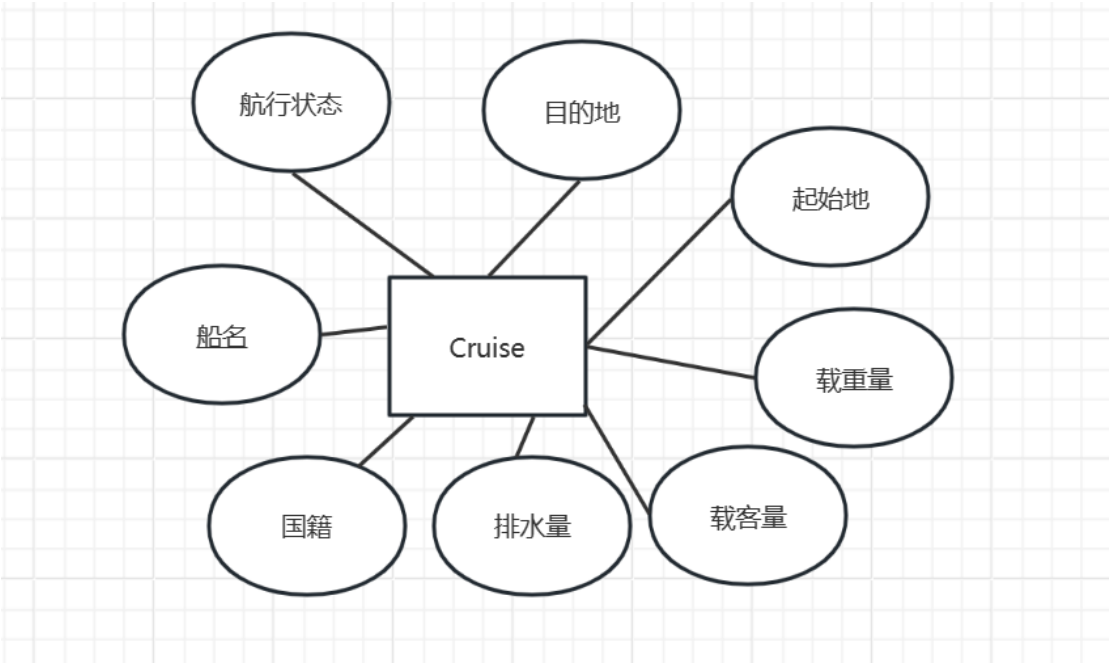
4.1 实体

我的数据库设计中主要有以下实体：
船只实体：Cruise
港口实体：Port
航线线路实体：CruiseLine
船长实体：Captain
港口管理员实体：Port Manager
系统管理员实体：System Manager
一般用户：Visitor
日志实体：Runlog

4.2 实体属性局部 E-R 图

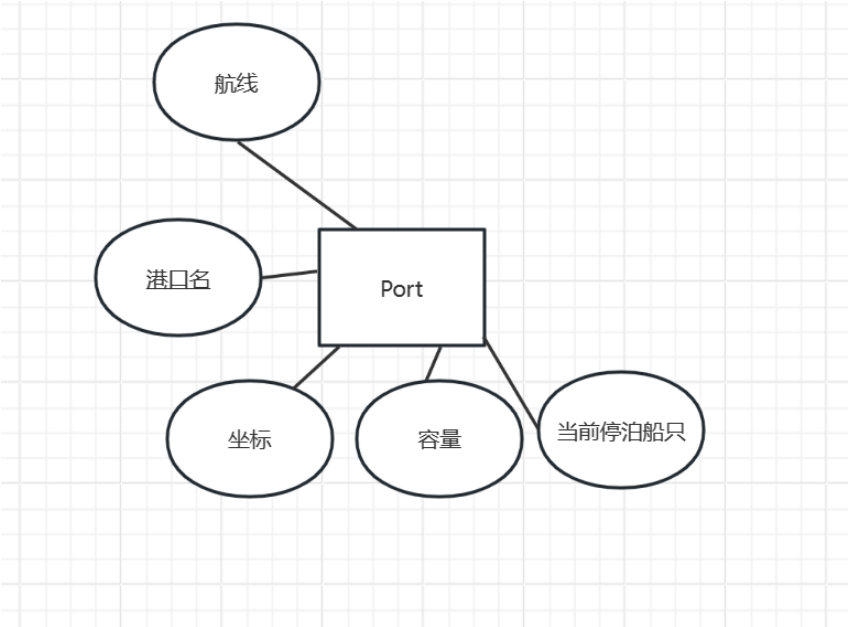
1. 船只实体：Cruise

船只实体用于记录正在航行中的船只信息。



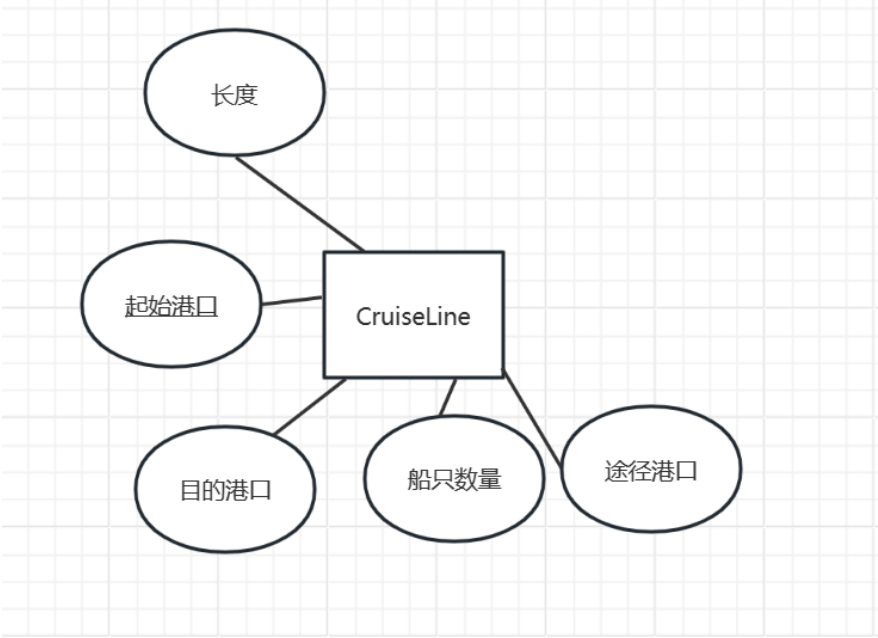
2. 港口实体：Port

港口实体用来记录城市中每个飞机场的详细信息。



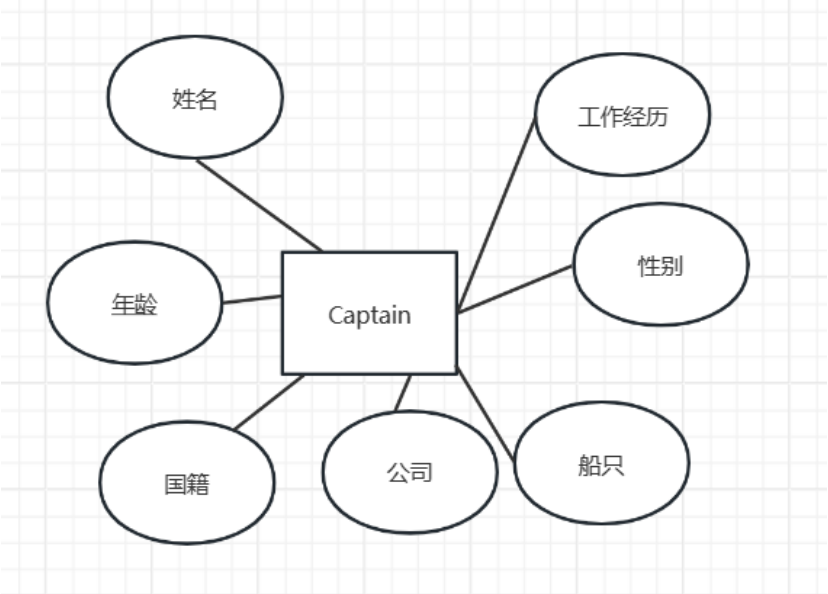
3. 航线线路实体：CruiseLine

航线线路实体用来存储线路信息。



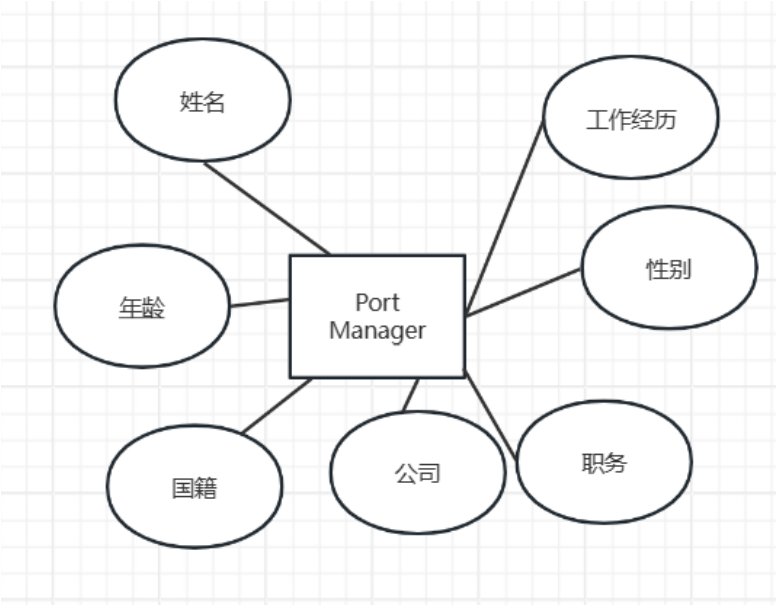
4. 船长实体：Captain

船长实体用来存储船长个人信息。



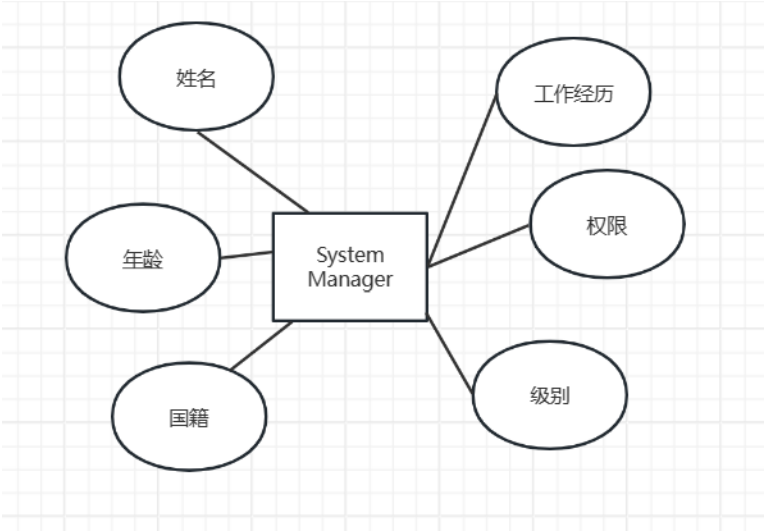
5. 港口管理员实体：Port Manager

港口管理员实体用来存储港口管理员个人信息。

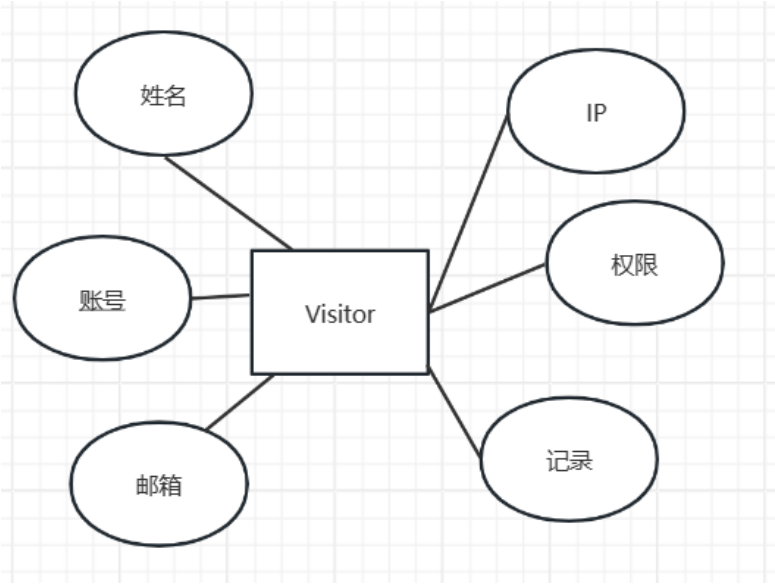


6. 系统管理员实体：System Manager

系统管理员实体用来存储系统管理员个人信息。

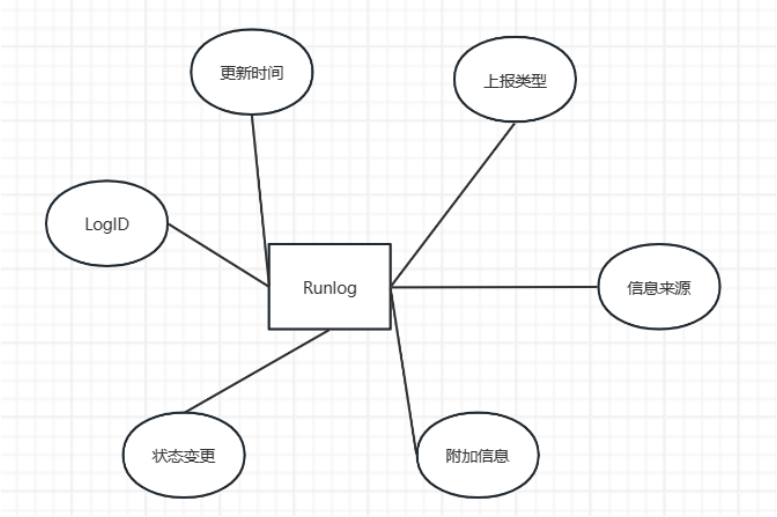


7. 一般用户



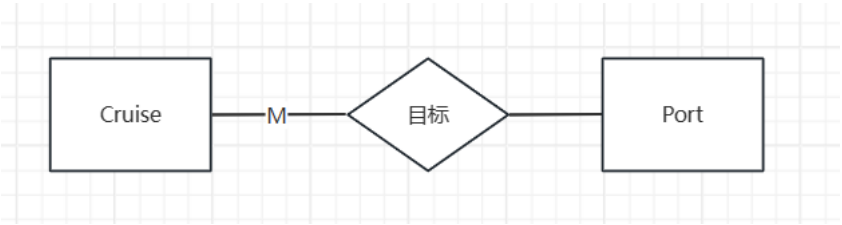
8. 日志实体：Runlog

日志实体用于处理各类实体的信息上报。

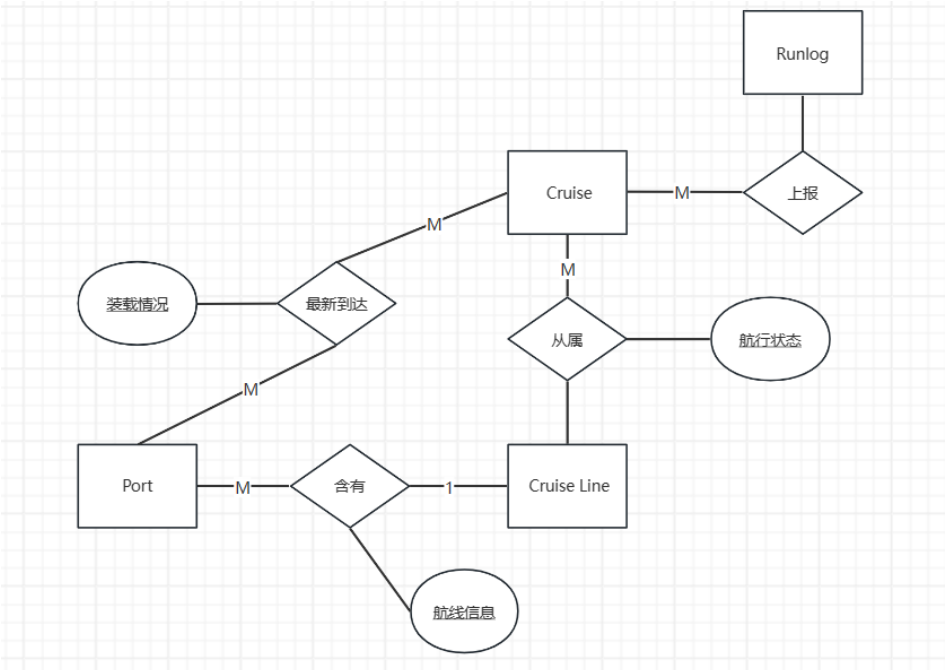


4.3 全局 E-R 图

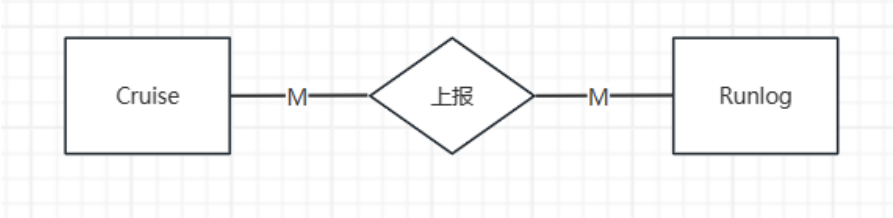
船只部分间实体-联系图：



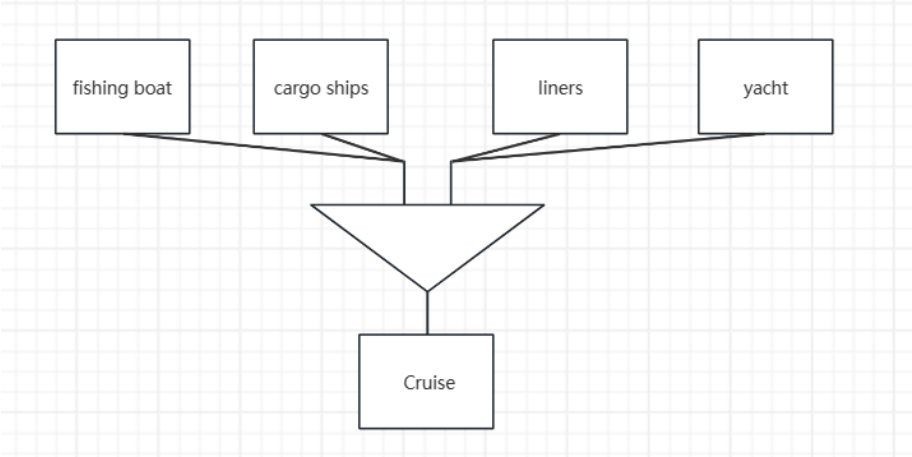
日志部分间实体-联系图：



船只-日志间实体图：



关系继承：



五.逻辑设计

5.1 E-R 图向关系模型的转变

根据上述 E-R 图，转换成关系模型如下：

1. 船只表（Ship）

Ship(ShipID, ShipName, Type)

2. 航行记录表（Voyage）

Voyage(VoyageID, ShipID, StartDate, EndDate)

3. 港口表（Port）

Port(PortID, PortName, Location)

4. 途经表（Passes）

Passes(VoyageID, PortID)

关系模型说明

1. 船只表（Ship）

Ship(ShipID, ShipName, Type)

ShipID: 主键，唯一标识每艘船只。

ShipName: 船只名称。

Type: 船只类型。

2. 航行记录表（Voyage）

Voyage(VoyageID, ShipID, StartDate, EndDate)

VoyageID: 主键，唯一标识每次航行记录。

ShipID: 外键，引用船只表中的 ShipID，表示是哪艘船进行的航行。

StartDate: 航行开始日期。

EndDate: 航行结束日期。

3. 港口表（Port）

Port(PortID, PortName, Location)

PortID: 主键，唯一标识每个港口。

PortName: 港口名称。

Location: 港口位置。

4. 途经表（Passes）

Passes(VoyageID, PortID)

VoyageID: 复合主键之一，引用航行记录表中的 VoyageID，表示航行的标识。

PortID: 复合主键之一，引用港口表中的 PortID，表示航行途经的港口。

数据库表的示例

1. 船只表（Ship）

ShipID	ShipName	Type
1	Titanic	Liner
2	Poseidon	Cargo

2. 航行记录表（Voyage）

VoyageID	ShipID	StartDate	EndDate
101	1	1912-04-10	1912-04-15
102	2	2023-01-01	2023-01-15

3. 港口表（Port）

PortID	PortName	Location
1	Southampton	UK
2	New York	USA

4. 途经表（Passes）

VoyageID	PortID

```
| 101       | 1       |
| 101       | 2       |
| 102       | 1       |
+---------+-------+
```

5.2 数据模型的优化及规范化设计

3NF 确保每个非主键字段不传递依赖于主键之外的字段。即每个非主键字段只依赖于主键，而不依赖于其他非主键字段。

基于以上设计原则，我们可以优化和规范化设计“Cruises Earth”项目的数据库模型。以下是经过优化和规范化设计后的结果：

3.1 船只表（Ship）

```
CREATE TABLE Ship (
          ShipID INTEGER PRIMARY KEY,
          ShipName TEXT NOT NULL,
          Type TEXT NOT NULL
      );
```

3.2 航行记录表（Voyage）

```
CREATE TABLE Voyage (
          VoyageID INTEGER PRIMARY KEY,
          ShipID INTEGER NOT NULL,
          StartDate DATETIME NOT NULL,
          EndDate DATETIME NOT NULL,
          FOREIGN KEY (ShipID) REFERENCES Ship(ShipID)
      );
CREATE INDEX idx_ship_id ON Voyage(ShipID);
```

3.3 港口表（Port）

```
CREATE TABLE Port (
          PortID INTEGER PRIMARY KEY,
          PortName TEXT NOT NULL,
          Location TEXT NOT NULL
      );
```

3.4 途经表（Passes）

```
CREATE TABLE Passes (
        VoyageID INTEGER,
        PortID INTEGER,
        PRIMARY KEY (VoyageID, PortID),
        FOREIGN KEY (VoyageID) REFERENCES Voyage(VoyageID),
        FOREIGN KEY (PortID) REFERENCES Port(PortID)
    );
```

六.项目管理

6.1 框架选择

1. Python

Python 是一种高级编程语言，以其简单、易读和易维护的语法著称。选择 Python 作为项目的主要编程语言有以下几个原因：

- **易学易用**：Python 的语法简洁直观，非常适合快速开发和原型设计，对于开发者来说，能够更快地上手并进行开发。
- **丰富的库和框架**：Python 拥有丰富的库和框架生态系统，能够快速实现各种功能需求。例如，Tkinter 用于 GUI 开发，SQLite3 用于数据库管理。
- **广泛应用**：Python 在数据处理、科学计算、网络开发、自动化等领域有广泛应用，且拥有庞大的社区支持，能够提供大量的资源和帮助。

2. SQLite3

SQLite3 是一个轻量级、嵌入式的关系型数据库管理系统。选择 SQLite3 作为数据库的主要原因包括：

- **零配置**：SQLite3 不需要单独的服务器进程，数据库存储在一个单一的文件中，使用非常方便，特别适合中小型项目。
- **高效**：SQLite3 在单用户环境中表现出色，能够快速处理 SQL 查询，对于“Cruises Earth”这样一个单机版的船只管理系统来说，性能完全能够满足需求。
- **Python 内置支持**：Python 标准库中自带了 SQLite3 模块，开发者可以直接使用，不需要额外安装和配置，简化了开发过程。

3. TkinterMapview

TkinterMapview 是一个基于 Tkinter 的地图视图工具包，能够在 Python GUI 应用中嵌入地图功能。选择 TkinterMapview 进行可视化的原因包括：

> **与 Tkinter 无缝集成**：Tkinter 是 Python 标准库中的 GUI 工具包，TkinterMapview 作为其扩展，能够无缝集成地图功能，适合需要地图展示功能的桌面应用。

- **多种地图显示模式**：TkinterMapview 支持普通模式、街景模式和遥感模式，能够提供多种视角下的地图展示，满足用户的不同需求。
- **易于使用**：TkinterMapview 使用简单，API 友好，开发者可以快速上手，实现地图展示和交互功能。

总结

选择 Python + SQLite3 + TkinterMapview 框架进行“Cruises Earth”项目开发，是基于以下几个方面的考虑：

- **快速开发**：Python 语言的易用性和高效性，使得开发者能够快速实现项目需求，并进行迭代和优化。
- **轻量级数据库**：SQLite3 作为轻量级数据库，配置简单、性能高效，非常适合单机版的船只管理系统。
- **地图可视化**：TkinterMapview 能够在 Tkinter 基础上实现多种模式的地图展示，提供直观的船只位置和轨迹显示功能。

通过这种框架选择，项目能够在保证开发效率的同时，实现高效的船只信息管理和直观的地图展示功能，为用户提供良好的使用体验。

6.2 开发平台

解释器：　Python 3.12.1

系统：　Windows 11

Requirements:

```
beautifulsoup4==4.12.3
customtkinter==5.2.2
```

```
geopy==2.4.1
pandas==2.2.2
Pillow==10.3.0
Requests==2.32.3
tkintermapview==1.29
```

七.系统实现

7.1 系统架构搭建

系统架构主要分成以下几层：

用户界面（GUI）

用户界面层是系统与用户交互的窗口，主要负责显示和输入数据，并提供直观的地图展示功能。本系统的用户界面采用了 Tkinter 和 TkinterMapview。

- **Tkinter**：Python 标准库中的 GUI 工具包，用于构建桌面应用的基本窗口和控件（如按钮、文本框、标签等）。
- **TkinterMapview**：一个扩展 Tkinter 的地图视图工具包，支持普通模式、街景模式和遥感模式的地图展示。用户可以通过地图界面查看船只的实时位置和历史轨迹。

应用逻辑层

应用逻辑层是系统的核心部分，负责处理用户请求、执行业务逻辑、与数据层交互并将结果返回给用户界面。

- **Python 脚本**：包含各种功能模块和业务逻辑的实现，例如船只信息管理、航行记录处理、路径规划和预测分析等。
- **业务逻辑实现**：包括数据验证、权限控制、逻辑运算等。确保用户的每个操作都能够正确执行并返回准确结果。

数据层

数据层负责管理和存储系统中的所有数据。选用 SQLite3 作为数据库管理系统，存储船只信息、港口信息、航行记录和相关的地理信息数据。

- **SQLite3 数据库**：一个轻量级、嵌入式的关系型数据库，存储在单一文件中，便于管理和部署。适合单机版应用，不需要额外的服务器配置。
- **数据管理与存储**：包括数据库表的设计、数据的增删改查操作，以及数据的备份和恢复功能。确保数据的完整性和一致性。

系统工作流程如下：

1. **船只信息管理**：用户通过 GUI 添加、删除、修改和查询船只信息。应用逻辑层处理用户请求并与数据库交互，返回处理结果。
2. **航行记录管理**：用户可以记录船只的航行信息，包括起始港口、途经港口和结束港口，以及航行的起止时间。系统通过应用逻辑层处理这些信息并存储在数据库中。
3. **地图展示**：用户通过 TkinterMapview 查看船只的实时位置和历史轨迹。地图展示包括普通模式、街景模式和遥感模式，提供多种视角查看船只信息。
4. **路径规划和分析**：系统根据用户需求提供路径规划和预测分析功能，帮助用户优化航行路线，提升管理效率。

总结：

“Cruises Earth”系统架构采用 Python 作为开发语言，结合 SQLite3 进行数据存储和管理，并通过 Tkinter 和 TkinterMapview 实现直观的用户界面和地图展示功能。整个系统架构清晰，分层明确，能够高效地管理和展示全球船只信息，满足用户的多样化需求。

这种架构设计不仅简化了开发和维护过程，还确保了系统的稳定性和可扩展性，为未来功能的扩展和升级提供了坚实的基础。

7.2 系统逻辑设计

“Cruises Earth”系统的主要功能是全球船只信息管理和展示。以下是对系统逻辑设计的详细介绍，涵盖系统的主要功能模块、数据流、以及与数据库的交互逻辑。

船只信息管理模块

功能描述：该模块负责船只基本信息的管理，包括船只的添加、删除、修改和查询操作。

逻辑流程：

1. **添加船只**：用户在 GUI 中输入船只信息（如船只 ID、船名、类型），点击“添加”按钮。
   - 应用逻辑层接收请求，验证数据的完整性和正确性。
   - 如果验证通过，将数据插入到 SQLite3 数据库中的 Ship 表。
   - 返回操作结果给用户界面，显示成功或失败信息。
2. **删除船只**：用户在 GUI 中选择要删除的船只，点击“删除”按钮。
   - 应用逻辑层接收请求，根据船只 ID 从数据库中的 Ship 表中删除对应记录。
   - 返回操作结果给用户界面，显示成功或失败信息。
3. **修改船只**：用户在 GUI 中选择要修改的船只，修改相关信息后点击“保存”按钮。
   - 应用逻辑层接收请求，验证修改后的数据。
   - 如果验证通过，更新数据库中的 Ship 表。
   - 返回操作结果给用户界面，显示成功或失败信息。
4. **查询船只**：用户在 GUI 中输入查询条件，点击“查询”按钮。
   - 应用逻辑层接收请求，根据条件从数据库中的 Ship 表中检索数据。
   - 将查询结果返回给用户界面，并显示在界面上。

航行记录管理模块

功能描述：该模块负责管理船只的航行记录，包括记录的添加、查询、修改和删除。

逻辑流程：

1. **添加航行记录**：用户在 GUI 中输入航行信息（如航行 ID、船只 ID、开始日期、结束日期），点击“添加”按钮。
   - 应用逻辑层接收请求，验证数据的完整性和正确性。
   - 如果验证通过，将数据插入到 SQLite3 数据库中的 Voyage 表。
   - 返回操作结果给用户界面，显示成功或失败信息。
2. **删除航行记录**：用户在 GUI 中选择要删除的航行记录，点击“删除”按钮。
   - 应用逻辑层接收请求，根据航行 ID 从数据库中的 Voyage 表中删除对应记录。
   - 返回操作结果给用户界面，显示成功或失败信息。

3. **修改航行记录**：用户在 GUI 中选择要修改的航行记录，修改相关信息后点击“保存”按钮。
   - 应用逻辑层接收请求，验证修改后的数据。
   - 如果验证通过，更新数据库中的 Voyage 表。
   - 返回操作结果给用户界面，显示成功或失败信息。
4. **查询航行记录**：用户在 GUI 中输入查询条件，点击“查询”按钮。
   - 应用逻辑层接收请求，根据条件从数据库中的 Voyage 表中检索数据。
   - 将查询结果返回给用户界面，并显示在界面上。

地图展示功能模块

功能描述：该模块负责船只位置和航行轨迹的地图展示。

逻辑流程：

1. **显示船只位置**：用户在 GUI 中选择一个船只，点击“显示位置”按钮。
   - 应用逻辑层接收请求，从数据库中的 Ship 表和 Voyage 表中获取船只的最新位置信息。
   - 调用 TkinterMapview 的 API，在地图上显示船只的当前位置。
2. **显示航行轨迹**：用户在 GUI 中选择一个航行记录，点击“显示轨迹”按钮。
   - 应用逻辑层接收请求，从数据库中的 Voyage 表和 Passes 表中获取航行轨迹信息。
   - 调用 TkinterMapview 的 API，在地图上绘制船只的航行路线。

数据层

数据描述：系统使用 SQLite3 数据库进行数据存储和管理。数据库表结构如下：

1. Ship 表
   - ShipID (PK)
   - ShipName
   - Type
2. Voyage 表
   - VoyageID (PK)
   - ShipID (FK)
   - StartDate
   - EndDate
3. Port 表
   - PortID (PK)

- PortName
- Location

4. Passes 表

- VoyageID (FK, Composite PK)
- PortID (FK, Composite PK)

总结

“Cruises Earth”系统通过清晰的逻辑设计，确保了船只信息和航行记录的高效管理，并通过直观的地图展示功能，为用户提供了良好的使用体验。采用 Python 进行应用逻辑处理，SQLite3 进行数据存储，以及 TkinterMapview 进行地图展示，使得整个系统架构简洁明了，易于开发和维护。

7.3 具体功能编写

下面展示一些核心代码段的实现：

数据库操作：

```
def insert_data():
    # Insert data into the database
    id = id_entry.get()
    price = price_entry.get()
    location = location_entry.get()
    ton=ton_entry.get()
    shipURL=shipURL_entry.get()
    shipname=shipname_entry.get()
    c.execute("INSERT INTO cruises VALUES (?, ?, ?, ?, ?, ?, ?)", (id,
price, location,'NA',ton,shipURL,shipname))
    conn.commit()

    # Show a success message
    messagebox.showinfo("Success", "Data inserted successfully")



# Function to delete data
def delete_data():
```

```
    # Delete data from the database
    id = id_entry.get()

    shipname=shipname_entry.get()

    c.execute("DELETE FROM cruises WHERE id = ?", (id,))
    c.execute("DELETE FROM cruises WHERE shipName = ?", (shipname,))
    conn.commit()

    # Show a success message
    messagebox.showinfo("Success", "Data deleted successfully")



# Function to lookup data
def lookup_data():
    # Lookup data from the database


    id = id_entry.get()
    shipname = shipname_entry.get()

    c.execute("SELECT * FROM cruises WHERE id = ?", (id,))
    c.execute("SELECT * FROM cruises WHERE shipName = ?", (shipname,))



    rows = c.fetchall()

    # Show the retrieved data
    for row in rows:
        print(row)
```

地图航线显示：

```
        tempArr = []
        for j in range(x):
            tempMarker = self.map_widget.set_address(a[j],
marker=True)   ##描点
            if tempMarker:
                if tempBool and not j:
                        tempStr = str(j + 1) + '-' + str(len(a)) +'. '+
str(a[j])
                        tempMarker.set_text(tempStr)
                        self.marker_list.append(tempMarker)
```

```
                        tempArr.append(tempMarker.position)
                else:
                    tempStr = str(j+1) + '. '+ str(a[j])
                    tempMarker.set_text(tempStr)
                    self.marker_list.append(tempMarker)
                    tempArr.append(tempMarker.position)

        tempPoly = app.map_widget.set_polygon(tempArr,
outline_color="red",
                                              border_width=12,
                                              name="switzerland_polygon")
 ##画线
```

前端从数据库中获取信息：

```
# Scrape the data from the local database
def scrape_cruise(tupleArr):
    tempArrTuples = []

    # Connect to the SQLite database
    conn = sqlite3.connect('cruise_info.db')
    c = conn.cursor()

    # Fetch data from the database
    c.execute("SELECT * FROM cruises")
    rows = c.fetchall()

    # Close the connection
    conn.close()

    # Process the data
    for row in rows:
        # Each row is a tuple (month, price, link)
        tempArrTuples.append(row)

    return tempArrTuples
```

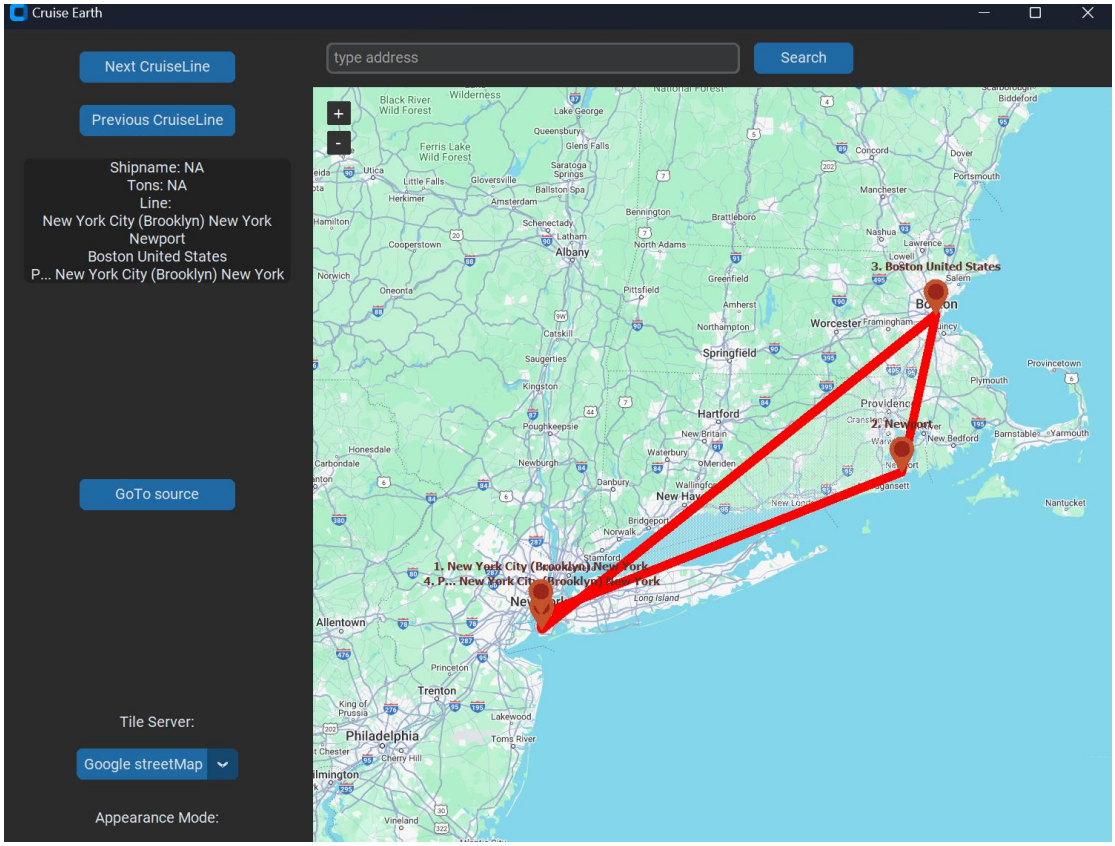
具体功能的开发过程日志如下：

```
## 开发日志

### 2024年4月29日
    实现数据库操作的main界面和三个子界面的完成，数据库操作还未测试

### 2024年6月1日
    基本功能已经全部开发完毕。
    还有这么几个小任务需要做。

    1、数据库查询显示结果能否不在命令行输出？
    2、数据库元素组合(id, Price, Location, Dates, ton, shipURL, shipName)，还没探索如何利用shipurl。但是七个元素不可能全部都手动输入
    3、完成2之后收集数据


## 2024年6月1日
    修改：
    1、 tkintermapview源从openstreetmap换到了更稳定的谷歌地图


## 2024年6月2日
    修改：
    1、不纠结地图源了，主要是网络的问题，更稳定的节点即可。
    2、完工！！


## 2024年6月15日
    增加：
    1、printall功能集成到数据库操作台
```

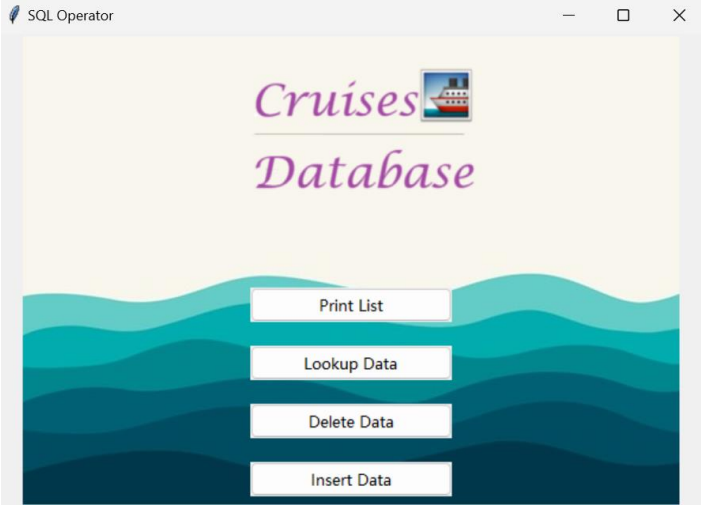
7.4 功能测试

具体的可以看演示视频。下面介绍一些关键功能的测试。

地图：



数据库操作台：



数据插入测试：

```
c.execute('''CREATE TABLE IF NOT EXISTS cruises
             (id, Price, Location, Dates, ton, shipURL, shipName)''')

(1,$199,Sydney;Brisbane,NA,92720,https://www.carnival.com/cruise-ships/carnival-luminosa, CARNIVAL LUMINOSA)
(2,$359,Los Angeles;Ensenada;Cabo San Lucas,NA,135156,https://www.carnival.com/cruise-ships/carnival-firenze, carnival firenze)
(3, 499, 'Miami;Cozumel;Key West', NULL, 110000, 'https://www.royalcaribbean.com/cruise-ships/symphony-of-the-seas', 'symphony
of the seas'),
(4, 699, 'Barcelona;Palma de Mallorca;Florence', NULL, 139000, 'https://www.ncl.com/cruise-ships/norwegian-epic', 'norwegian
epic'),
(5, 399, 'Galveston;Cozumel;Progreso', NULL, 128000, 'https://www.carnival.com/cruise-ships/carnival-breeze', 'carnival
breeze'),
(6, 599, 'New York;San Juan;St. Thomas', NULL, 169379, 'https://www.royalcaribbean.com/cruise-ships/anthem-of-the-seas',
'anthem of the seas'),
(7, 549, 'Seattle;Juneau;Skagway', NULL, 116000, 'https://www.hollandamerica.com/cruise-ships/ms-noordam', 'ms noordam'),
(8, 799, 'Fort Lauderdale;Nassau;CocoCay', NULL, 168666, 'https://www.royalcaribbean.com/cruise-ships/freedom-of-the-seas',
'freedom of the seas'),
(9, 459, 'San Francisco;Victoria;Vancouver', NULL, 142000, 'https://www.princess.com/cruise-ships/majestic-princess', 'majestic
princess'),
(10, 629, 'Miami;Grand Cayman;Jamaica', NULL, 133500, 'https://www.carnival.com/cruise-ships/carnival-vista', 'carnival vista'),
(11, 679, 'Sydney;Auckland;Wellington', NULL, 167800, 'https://www.celebritycruises.com/cruise-ships/celebrity-eclipse',
'celebrity eclipse');
```

八.总结

完成“Cruises Earth”项目让我感受到了从构思到实现一个完整系统的成就感。在这个过程中，我不仅加深了对技术的理解，也体会到了项目管理和团队协作的重要性。以下是我对这个项目的一些感想：

在这个项目中，我使用了 Python、SQLite3 和 TkinterMapview，全面提升了我的编程技能和技术应用能力。通过这个项目，我熟练掌握了如何使用 Python 进行快速开发，如何高效地管理和查询 SQLite3 数据库，以及如何通过 TkinterMapview 进行直观的地图展示。

从项目的需求分析、系统设计到最终的实现和测试，每一个环节都需要细致的规划和执行。通过这个项目，我深刻理解了项目管理的重要性。

虽然这是一个个人项目，但在一些关键环节，我也借助了同事和朋友的帮助与建议。这里特别感谢贺松同学。他们的反馈帮助我改进了系统设计和用户体验，理解了团队协作的重要性。

完成“Cruises Earth”项目让我在技术、项目管理和用户体验等方面都有了全面的提升。这个项目不仅是对我技术能力的一次检验，也是对我综合素质的一次提升。最后感谢李文根老师对我项目提出的意见！

参考文献和链接

[1]. Tournadre, Jean. "Anthropogenic pressure on the open ocean: The growth of ship traffic revealed by altimeter data analysis." *Geophysical Research Letters* 41.22 (2014): 7924-7932.

[2]. Tournadre, Jean. "Anthropogenic pressure on the open ocean: The growth of ship traffic revealed by altimeter data analysis." *Geophysical Research Letters* 41.22 (2014): 7924-7932.

[3]. TomSchimansky/TkinterMapView: A python Tkinter widget to display tile based maps like OpenStreetMap or Google Satellite Images. (github.com)

[4]. Levitus, Sydney, et al. "The world ocean database." *Data Science Journal* 12 (2013): WDS229-WDS234.

[5]. Xu, Sangming, Zhiliang Gao, and Wen Xue. "CFD database method for roll response of damaged ship during quasi-steady flooding in beam waves." *Applied Ocean Research* 126 (2022): 103282.

[6]. Tournadre, J., and Jean-François Piollé. "The ship database version-3.1 (1992-2022)." (2023).

[7]. Koyama, Takeo, Hiroyuki Yamato, and Shigehisa Betchaku. "A study on the design system based on the ship database." *Journal of the Society of Naval Architects of Japan* 2014.176 (2014): 543-550.

[8]. Ou, Ziqiang, and Jianjun Zhu. "AIS database powered by GIS technology for maritime safety and security." *The Journal of Navigation* 61.4 (2008): 655-665.