海洋表面温度数据库系统设计与开发报告

2151140 王谦 信息安全 授课老师：李文根

摘要：随着全球气候变化对人类社会的影响日益加剧，海洋表面温度（SST）作为关键的环境参数，其数据的收集、管理和分析对于气候研究、环境监测和资源管理具有至关重要的作用。然而，现有的海洋表面温度数据管理系统多集中于基础数据展示，缺乏深入分析和个性化查询功能，难以满足科研人员和决策者的具体需求。鉴于此，本研究设计并开发了一个综合性的海洋表面温度数据库系统，旨在提供更为丰富和灵活的数据服务。

本系统通过需求分析，明确了用户对数据输入、多条件查询、数据筛选、导出功能以及用户和数据管理的具体需求。系统采用模块化设计，利用C#和.NET框架结合SQL Server数据库，构建了一个高效、稳定且易于维护的客户端-服务器架构。概念设计阶段，通过E-R图转化为关系模型，确保了数据模型的规范化和优化。在项目管理方面，选用了WinForms作为前端框架，Entity Framework作为ORM工具，并采用Visual Studio作为开发环境，以提高开发效率和代码质量。

系统实现包括用户友好的登录与注册机制、权限管理、数据展示、动态查询、数据筛选和导出等功能。特别是针对当前环境变化和政策导向，系统增加了对数据的深入分析和可视化展示，以及对数据源的实时监控和质量控制，确保数据的准确性和可靠性。通过功能测试，验证了系统设计的合理性和实现的正确性。

本研究不仅为海洋表面温度数据的管理和分析提供了一个高效、准确、易用的解决方案，而且通过不断的技术创新和功能完善，满足了用户对数据深度分析和个性化服务的需求。系统的成功实施，将有助于科研人员和决策者更好地理解和应对气候变化带来的挑战，同时也为相关领域的技术进步和知识创新提供了支持。

关键词：海洋表面温度数据库；数据管理；系统设计；气候变化；环境监测

一.概述

1.1课题背景

海洋表面温度（Sea Surface Temperature，SST）是指海洋表层的一定深度范围内的水温。它是气候系统中一个重要的变量，直接影响到天气和气候变化。SST的变化可以引发和调节各种大气现象，如厄尔尼诺现象和拉尼娜现象，进而影响全球气候模式。因此，海洋表面温度的监测和研究对于气候预测、环境保护和海洋资源管理具有重要意义。

随着科学技术的发展，海洋观测技术日益先进，获取的海洋数据量大幅增加。建立一个高效、准确、易用的海洋表面温度数据库，可以为科研工作者、环境保护机构和海洋管理部门提供可靠的数据支持，从而更好地理解和应对气候变化的挑战。

1.2 编写目的

本数据库课程设计的主要目的是开发一个海洋表面温度数据库系统，旨在实现以下目标：

1. **数据存储与管理**：有效存储和管理大量的海洋表面温度数据，确保数据的完整性和一致性。
2. **数据查询与筛选**：提供便捷的查询功能，用户可以根据经度、纬度、深度、时间等条件进行数据筛选，满足不同的研究需求。
3. **数据分析与展示**：支持数据的分析和展示，如数据的导出功能，可以将查询结果导出为CSV文件，以便进行进一步的分析。
4. **用户友好界面**：提供简洁易用的用户界面，使用户能够方便地进行数据的查询、筛选和导出操作。

1.3 用户需求

根据对潜在用户（包括科研人员、环境保护机构和海洋管理部门）的调研，系统需要满足以下具体需求：

1. **数据输入**：能够从数据源中导入海洋表面温度数据，并进行数据的验证和清洗，确保数据的准确性。
2. **多条件查询**：用户可以通过经度、纬度、深度和时间等多种条件进行数据查询，并能够处理用户只填写部分条件的情况，提供灵活的查询功能。
3. **数据筛选**：用户可以在查询结果中进一步筛选数据，如指定经纬度范围、深度范围和时间范围，以便获取更加精确的数据。
4. **数据导出**：查询结果可以导出为CSV文件，便于用户进行离线分析和共享数据。
5. **用户管理**：普通用户可以注册新账号和查看数据，管理员可以管理用户账号（如添加、删除、修改用户信息）和修改海洋表面温度数据。
6. **用户界面**：界面简洁明了，操作简单，用户能够方便地进行数据查询、筛选和导出操作，提升用户体验。

二.需求分析

2.1 功能需求

用户管理：

- **注册和登录**：普通用户可以注册新账号，登录系统查看数据。管理员可以登录后进行数据管理和用户管理。
- **用户账号管理**：管理员可以添加、删除和修改用户信息，分配和管理用户权限。

管理界面：包括数据查询、筛选、显示和导出功能，用户可以在主界面上输入查询条件并查看结果。

数据筛选：通过文本框输入筛选条件，如经度、纬度、深度和时间，支持多条件组合筛选，用户可以按需调整筛选条件。

数据查询：通过文本框输入查询条件，如经度、纬度、深度和时间，实现准确的多条件查询。

数据导出：提供将查询结果导出为CSV文件的功能，用户可以选择保存路径，并导出数据进行离线分析。

2.2 数据字典

用户表（t_user）

属性名	类型	约束	描述
UserID	INT	PRIMARY KEY	用户的唯一标识符
Username	VARCHAR(255)	NOT NULL	用户名
Password	VARCHAR(255)	NOT NULL	密码
Email	VARCHAR(255)	NOT NULL	电子邮件地址
UserRole	VARCHAR(255)	NOT NULL	身份备注

管理员表（t_admin）

属性名	类型	约束	描述
AdminID	INT	PRIMARY KEY	管理员的唯一标识符
Username	VARCHAR(255)	NOT NULL	用户名
Password	VARCHAR(255)	NOT NULL	密码
Email	VARCHAR(255)	NOT NULL	电子邮件地址
UserRole	VARCHAR(255)	NOT NULL	身份备注

温度表（t_temp）

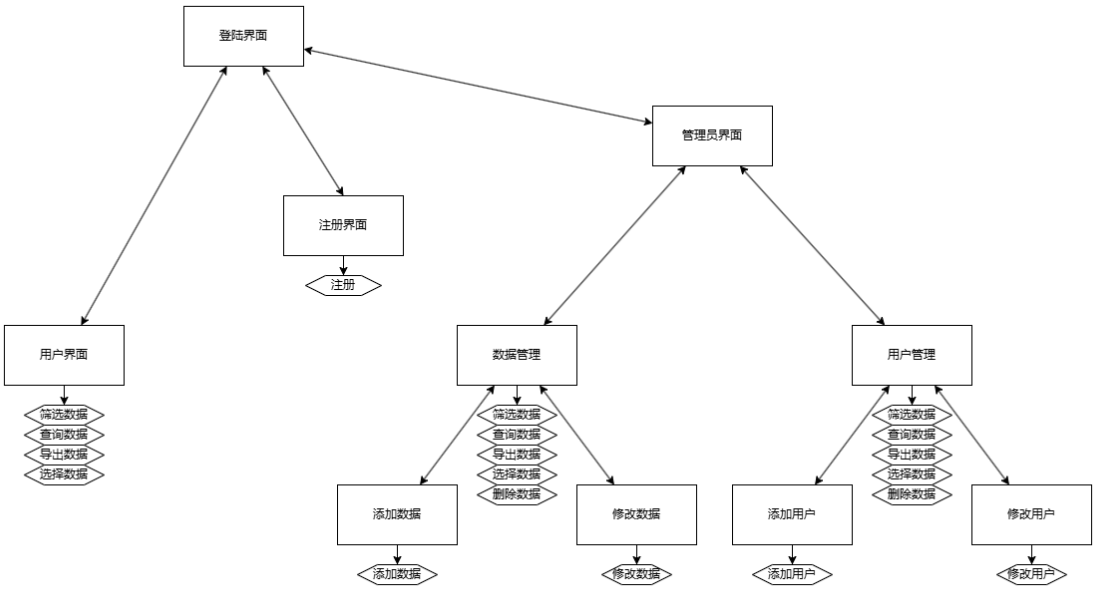
属性名	类型	约束	描述
TempID	INT	PRIMARY KEY	温度记录的唯一标识符
Lon	FLOAT	NOT NULL	经度
Lat	FLOAT	NOT NULL	纬度
Depth	FLOAT	NOT NULL	深度
Time	INT	NOT NULL	时间
Temperature	FLOAT		温度

2.3 数据流图

数据流图说明

1. 用户和管理员通过各自的用户界面与系统交互。
2. 用户进行登录、注册、查看数据和筛选数据的操作。
3. 管理员额外拥有管理用户账号和修改数据的权限。
4. 所有用户操作通过应用程序发送到数据库进行处理。

数据流图



数据流图详细描述

1. **用户界面子系统**：
   - **用户登录/注册处理**：处理用户的登录和注册请求，验证用户身份，创建新用户账户。
   - **用户数据查询和筛选处理**：用户可以查询和筛选海洋表面温度数据，进行条件筛选和范围筛选。
   - **导出数据**：用户可以将查询和筛选的数据导出为CSV文件。
2. **管理员界面子系统**：
   - **管理员登录处理**：处理管理员的登录请求，验证管理员身份。
   - **用户管理处理**：管理员可以查看、添加、删除和修改用户信息。
   - **数据管理处理**：管理员可以查看、添加、删除和修改海洋表面温度数据。
   - **导出数据**：管理员可以将数据导出为CSV文件。

详细流程描述

用户登录/注册处理

1. 用户通过用户界面输入用户名和密码，选择登录或注册。
2. 系统验证用户身份，如果是登录则查询用户表验证用户名和密码是否匹配；如果是注册则将新用户信息插入用户表。
3. 验证通过后，用户进入用户界面；如果验证失败，则提示错误信息。

用户数据查询和筛选处理

1. 用户通过用户界面输入查询条件或筛选范围。
2. 系统根据用户输入的条件查询温度数据表，返回符合条件的数据。
3. 用户界面显示查询和筛选结果。

导出数据

1. 用户或管理员在用户界面或管理员界面选择导出数据。
2. 系统将当前显示的表格数据导出为CSV文件。
3. 用户或管理员下载CSV文件。

管理员登录处理

1. 管理员通过管理员界面输入用户名和密码，选择登录。
2. 系统验证管理员身份，查询管理员表验证用户名和密码是否匹配。
3. 验证通过后，管理员进入管理员界面；如果验证失败，则提示错误信息。

用户管理处理

1. 管理员在管理员界面查看用户信息。
2. 管理员可以选择添加、删除或修改用户信息。
3. 系统根据管理员的操作更新用户表。

数据管理处理

1. 管理员在管理员界面查看海洋表面温度数据。
2. 管理员可以选择添加、删除或修改温度数据。
3. 系统根据管理员的操作更新温度数据表。

三.可行性分析

3.1 技术可行性

开发海洋表面温度数据库系统的技术可行性分析包括以下几个方面：

1. **开发环境和工具**：
   - **编程语言**：本系统选择使用C#和.NET框架进行开发，这些技术已经非常成熟，广泛应用于各类企业级应用开发，具有良好的社区支持和丰富的第三方库资源。
   - **数据库管理系统**：选用SQL Server作为后台数据库管理系统，SQL Server具有高效的数据处理能力，支持复杂的查询操作和事务处理，能够满足本系统的需求。
   - **开发工具**：使用Visual Studio作为集成开发环境（IDE），它提供了强大的调试和开发工具，支持快速开发和部署。
2. **系统架构**：
   - **客户端-服务器架构**：采用典型的客户端-服务器架构，客户端通过用户界面与用户交互，服务器负责处理业务逻辑和数据库操作。该架构能够有效分离前端和后端，增强系统的扩展性和维护性。
   - **模块化设计**：系统功能模块化设计，包括用户管理模块、数据查询模块、数据导出模块等，便于系统的开发和维护。
3. **数据存储和处理**：
   - **数据量和存储**：海洋表面温度数据的存储需求较大，SQL Server能够高效地处理和存储大量数据，并提供完善的数据备份和恢复机制。
   - **数据查询和筛选**：通过SQL Server的强大查询功能，能够实现复杂的多条件查询和数据筛选，满足用户的多样化需求。
4. **安全性和权限管理**：
   - **用户权限控制**：系统实现了管理员和普通用户的权限分离，确保只有管理员能够管理用户账号和修改数据，增强系统的安全性。
   - **数据安全**：通过SQL Server的安全机制，确保数据在存储和传输过程中的安全性，防止未经授权的访问和修改。

3.2 应用可行性

应用可行性分析主要考虑系统的实用性和用户接受度：

1. **用户需求满足**：
   - **科研人员**：系统提供的多条件查询和数据筛选功能能够满足科研人员对海洋表面温度数据的分析需求。数据导出和可视化功能也为科研人员的进一步研究提供了便利。
   - **环境保护机构**：通过系统，可以方便地获取和分析海洋表面温度数据，支持环境保护决策。数据管理和用户权限控制功能确保数据的准确性和安全性。
   - **海洋管理部门**：系统能够帮助海洋管理部门监测和管理海洋表面温度数据，提供决策支持。用户管理功能便于部门内部人员的权限分配和管理。
2. **系统易用性**：
   - **用户界面**：系统提供简洁明了的用户界面，用户能够方便地进行数据查询、筛选和导出操作。对于普通用户，系统提供注册和查看数据的功能，界面友好，操作简单。
   - **系统响应速度**：SQL Server和.NET框架的高效处理能力保证了系统在处理大量数据时的响应速度，用户体验良好。
3. **系统推广和维护**：
   - **培训和支持**：系统开发完成后，可以为用户提供必要的培训，帮助他们熟悉系统的使用。同时，提供技术支持和维护服务，确保系统的长期稳定运行。
   - **扩展性和可维护性**：系统采用模块化设计，便于后续功能的扩展和维护。用户反馈的需求和建议可以快速在系统中实现和更新。

综上所述，通过技术可行性和应用可行性的分析，可以确认开发海洋表面温度数据库系统在技术上是可行的，并且能够有效满足用户的实际需求。系统具有良好的实用性和用户接受度，能够为科研工作、环境保护和海洋管理提供有力支持。

四.概念设计

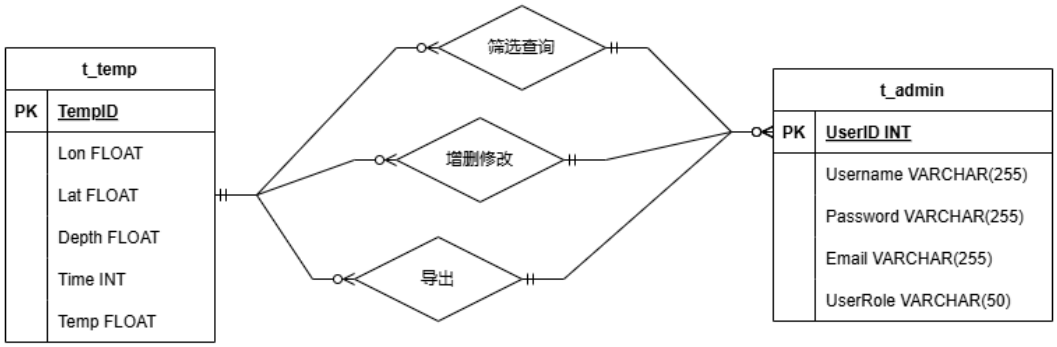
4.1 实体

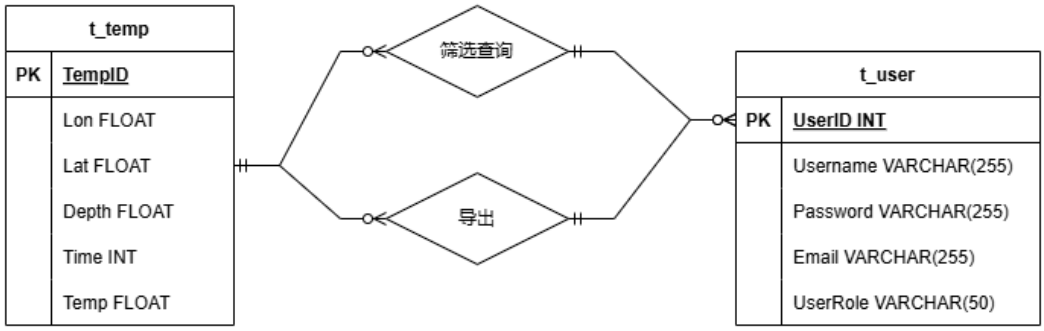
在海洋表面温度数据库系统中，主要有以下几个实体：

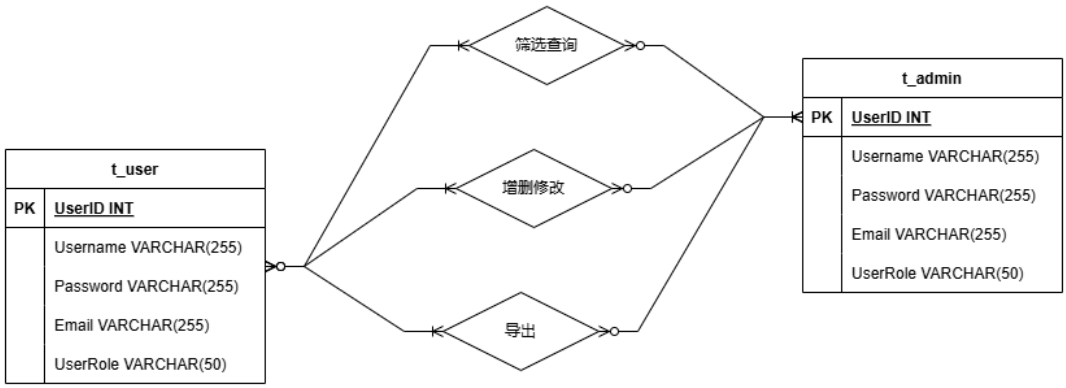
1. **用户（User）**：
   - 描述：系统中使用的用户。
   - 属性：
     - UserID (INT)：用户的唯一标识符
     - Username (VARCHAR(255))：名称
     - Password (VARCHAR(255))：用户密码
     - Email (VARCHAR(255))：电子邮件地址
     - UserRole (VARCHAR(50))：身份备注
2. **管理员（Admin）**：
   - 描述：系统中负责管理的用户。
   - 属性：
     - AdminID (INT)：管理员的唯一标识符
     - Username (VARCHAR(255))：称呼
     - Password (VARCHAR(255))：用户密码
     - Email (VARCHAR(255))：电子邮件地址
3. **温度数据（Temperature）**：
   - 描述：记录海洋表面温度的数据。

- 属性：
  - TempID (INT)：温度记录的唯一标识符
  - Lon (FLOAT)：经度
  - Lat (FLOAT)：纬度
  - Depth (FLOAT)：深度
  - Time (INT)：时间
  - Temp (FLOAT)：温度

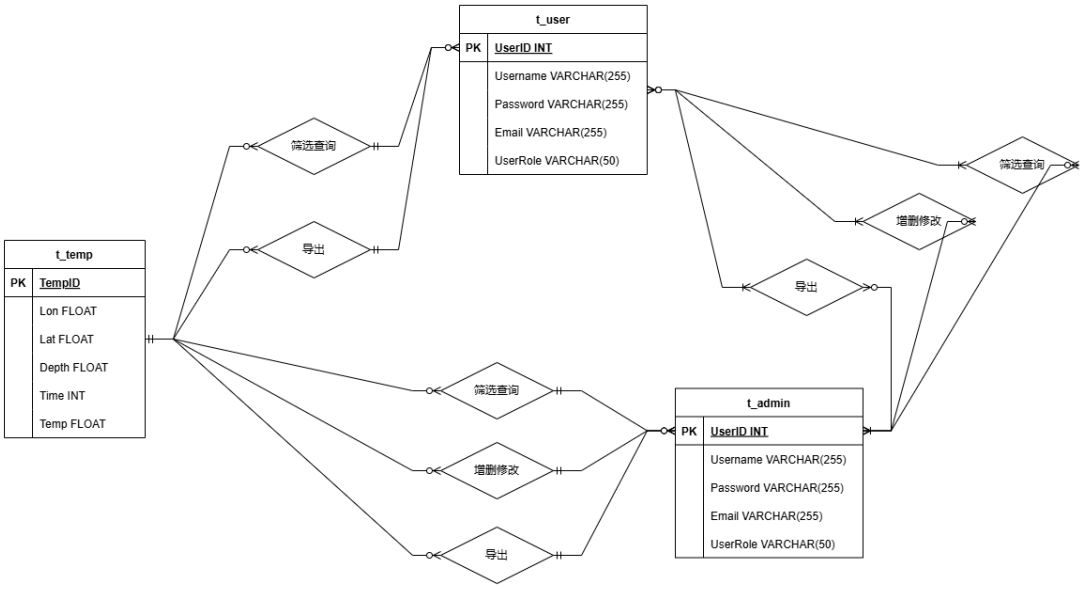
4.2 实体属性局部E-R图







4.3全局E-R图



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，将其转化为关系模型。

1. **用户表 (User)**:
    - UserID: INT, PRIMARY KEY
    - Username: VARCHAR(255)
    - Password: VARCHAR(255)
    - Email: VARCHAR(255)
    - UserRole: VARCHAR(50)
2. **管理员表 (Admin)**:
    - AdminID: INT, PRIMARY KEY
    - Username: VARCHAR(255)
    - Password: VARCHAR(255)
    - Email: VARCHAR(255)
3. **温度数据表 (Temperature)**:
    - TempID: INT, PRIMARY KEY
    - Lon: FLOAT
    - Lat: FLOAT
    - Depth: FLOAT
    - Time: INT
    - Temperature: FLOAT

5.2 数据模型的优化及规范化设计

数据模型的优化及规范化设计是为了减少数据冗余和数据异常，提高数据一致性。

1. **第一范式 (1NF)**：确保每列都是不可分割的原子值。
    - 所有表中的每个字段都包含单一值，不存在多值字段。
2. **第二范式 (2NF)**：消除非主属性对码的部分函数依赖。
    - 温度数据表中，TempID是主键，其他字段完全依赖于TempID，不存在部分依赖。
3. **第三范式 (3NF)**：消除非主属性对码的传递函数依赖。
    - 所有表中的非主属性仅依赖于主键，没有传递依赖。

通过上述规范化过程，确保了数据库的结构合理，数据存储高效且一致性高。

六.项目管理

6.1 框架选择

在选择开发框架时，需要考虑系统的功能需求、性能要求、开发的技术栈以及未来的扩展性和维护性。对于海洋表面温度数据库系统，以下是推荐的框架和理由：

1. **前端框架**：
    - **WinForms**：由于系统的主要用户界面是桌面应用程序，选择WinForms作为前端框架。WinForms是.NET框架的一部分，提供了丰富的控件和强大的事件处理机制，能够快速构建功能齐全、用户友好的桌面应用程序。
2. **后端框架**：
    - **.NET Framework**：选择.NET Framework作为后端框架。它与WinForms高度集成，支持C#编程语言，并且具有良好的性能和稳定性。使用.NET Framework可以方便地进行数据库连接、数据处理和业务逻辑实现。
    - **Entity Framework**：推荐使用Entity Framework进行数据访问和操作。Entity Framework是一个对象关系映射（ORM）框架，可以简化数据库操作，减少手动编写的SQL语句，提高开发效率。
3. **数据库管理系统**：
    - **SQL Server**：选择SQL Server作为数据库管理系统。SQL Server具有高效的数据存储和处理能力，支持复杂的查询操作和事务处理，能够满足系统对大量海洋表面温度数据的管理需求。
4. **数据导出和处理库**：
    - **ClosedXML**：用于将数据导出为Excel文件。ClosedXML是一个开源的.NET库，能够方便地创建、读取和操作Excel文件，支持丰富的Excel功能。
    - **CsvHelper**：用于将数据导出为CSV文件。CsvHelper是一个高性能的.NET库，能够快速、方便地处理CSV文件，支持多种配置选项和格式。

6.2 开发平台

选择合适的开发平台对于项目的顺利进行至关重要。考虑到开发环境、工具支持和团队技术栈，推荐使用以下开发平台：

1. **集成开发环境（IDE）**：
    - **Visual Studio**：选择Visual Studio作为主要的开发工具。Visual Studio是功能强大的集成开发环境，提供丰富的开发、调试和测试工具，支持多种编程语言和框架。它与.NET框架和WinForms高度集成，能够显著提高开发效率和代码质量。
2. **版本控制系统**：

- **Git**：使用Git进行版本控制和代码管理。Git是目前最流行的分布式版本控制系统，能够方便地进行代码分支管理、合并和协作开发。推荐使用GitHub或GitLab作为代码托管平台，便于团队成员之间的协作和项目管理。

七.项目实现

7.1 项目架构搭建

项目结构

1. **OceanSurfaceTemperatureDB**：主项目文件夹
   - **OceanSurfaceTemperatureDB.sln**：解决方案文件
   - **Dao.cs**：数据访问层，处理与数据库的连接和操作
   - **LoginData.cs**：存储用户登录信息的静态类
   - **UserWin.cs**：普通用户界面，显示数据表格和查询功能
   - **AdminWin.cs**：管理员主界面，包含系统管理和统计功能入口
   - **AdminWin1.cs**：管理员系统管理界面，处理用户管理等功能
   - **AdminWin11.cs**：管理员添加用户界面，处理数据插入功能
   - **AdminWin12.cs**：管理员修改界面，处理数据更新功能
   - **AdminWin2.cs**：管理员统计界面，显示数据统计和导出功能
   - **AdminWin21.cs**：管理员添加数据界面，处理数据插入功能
   - **AdminWin22.cs**：管理员修改数据界面，处理数据更新功能

技术选型

- **语言**：C#（基于.NET Framework）
- **数据库**：SQL Server（使用ADO.NET连接）
- **界面**：Windows Forms（用于构建用户界面）

架构设计

- **MVC模式**：
  - **Model**：数据库实体（例如 `t_temp` 表）、数据访问层（`Dao.cs`）
  - **View**：Windows Forms 窗体（`UserWin.cs`、`AdminWin.cs` 等）
  - **Controller**：业务逻辑和事件处理（窗体中的按钮点击事件、数据加载逻辑等）

7.2 系统逻辑设计

用户登录与权限管理

- **LoginData.cs** 存储用户登录信息，包括用户名和用户ID。
- **登录功能**：通过输入用户名和密码验证用户身份，登录成功后根据用户角色（管理员或普通用户）展示不同的界面。
- **注册功能**：新创建用户账号，并可以根据注册的账号登录用户界面。

普通用户功能

1. **数据显示与查询**：
   - 显示海洋表面温度数据（`t_temp` 表数据），支持根据条件查询（经度、纬度、时间等）。
   - 点击表格行显示详细数据。
2. **数据导出**：
   - 将当前显示的数据导出为CSV文件。

管理员功能

1. **系统管理**：
   - 用户管理：添加、修改、删除用户信息。
   - 系统设置：管理数据库连接、配置等。
2. **数据统计与管理**（AdminWin1.cs）：
   - 显示用户数据统计信息。
   - 增删修改用户账号。
   - 数据导出：将当前显示的数据导出为CSV文件。
3. **添加与修改数据**：
   - 添加数据：管理员可以手动输入用户数据并添加到数据库。
   - 修改数据：管理员可以选择某条数据进行修改操作。
4. **数据统计与管理**：
   - 显示海洋表面温度数据统计信息。
   - 支持按条件查询数据并展示在表格中。
   - 增删修改数据。
   - 数据导出：将当前显示的数据导出为CSV文件。
5. **添加与修改数据**：
   - 添加数据：管理员可以手动输入海洋表面温度数据并添加到数据库。
   - 修改数据：管理员可以选择某条数据进行修改操作，包括更新经度、纬度、深度、时间和温度信息。

7.3 具体功能编写

1. **Dao.cs**：封装了与数据库的连接和操作，包括执行SQL查询、更新、插入和删除操作，以及释放资源的方法：

```
class Dao : IDisposable
    {
        private readonly SqlConnection _connection;

        public Dao()
        {
            string connectionString = @"Data Source="";Initial
Catalog=OceanDB;Integrated Security=True;";
            _connection = new SqlConnection(connectionString);
            _connection.Open();
        }

        public SqlCommand CreateCommand(string sql)
        {
            return new SqlCommand(sql, _connection);
        }

        public SqlDataReader ExecuteReader(string sql)
        {
            SqlCommand cmd = CreateCommand(sql);
            return cmd.ExecuteReader();
        }

        public void Dispose()
```

```
        {
            if (_connection != null && _connection.State ==
System.Data.ConnectionState.Open)
            {
                _connection.Close();
            }
        }
    }
```

2. 数据表格的显示和更新功能：

```
private void Table()
        {
            dataGridView1.Rows.Clear(); // 清空旧数据
            string sql = "SELECT * FROM t_temp";
            using (SqlDataReader reader = dao.ExecuteReader(sql))
            {
                while (reader.Read())
                {
                    dataGridView1.Rows.Add(
                        reader[0].ToString(),
                        reader[1].ToString(),
                        reader[2].ToString(),
                        reader[3].ToString(),
                        reader[4].ToString(),
                        reader[5].ToString());
                }
            }
        }
```

3. 数据查询功能，支持动态条件查询：

```
            dataGridView1.Rows.Clear(); // 清空旧数据
            // 构建基本的查询语句
            StringBuilder sql = new StringBuilder("SELECT * FROM t_temp
WHERE 1=1");
            // 动态添加用户填写的条件
            if (!string.IsNullOrEmpty(textBox13.Text))
            {
                sql.Append($" AND TempID >= {textBox13.Text}");
            }
            if (!string.IsNullOrEmpty(textBox14.Text))
            {
                sql.Append($" AND TempID <= {textBox14.Text}");
            }
            ......
            // 执行查询
            using (SqlDataReader reader = dao.ExecuteReader(sql.ToString()))
            {
                while (reader.Read())
                {
```

```
                    dataGridView1.Rows.Add(
                        reader[0].ToString(),
                        reader[1].ToString(),
                        reader[2].ToString(),
                        reader[3].ToString(),
                        reader[4].ToString(),
                        reader[5].ToString());
                }
            }
```

4. 数据导出功能，将当前显示的数据导出为CSV文件：

```
private void ExportToCsv()
        {
            // 提示用户选择保存路径
            using (SaveFileDialog sfd = new SaveFileDialog() { Filter = "CSV
文件|*.csv", FileName = "DataExport.csv" })
            {
                if (sfd.ShowDialog() == DialogResult.OK)
                {
                    // 创建文件流
                    using (StreamWriter sw = new StreamWriter(sfd.FileName,
false, Encoding.UTF8))
                    {
                        // 写入列标题
                        var columnHeaders =
dataGridView1.Columns.Cast<DataGridViewColumn>();
                        sw.WriteLine(string.Join(",",
columnHeaders.Select(column => column.HeaderText).ToArray()));

                        // 写入行数据
                        foreach (DataGridViewRow row in dataGridView1.Rows)
                        {
                            if (!row.IsNewRow)
                            {
                                var cells = row.Cells.Cast<DataGridViewCell>
();
                                sw.WriteLine(string.Join(",",
cells.Select(cell => cell.Value?.ToString()).ToArray()));
                            }
                        }
                    }

                    MessageBox.Show("数据已成功导出！", "导出完成",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }
```

5. 导出nc文件数据的matlab脚本：

```
% 文件路径
file_path = 'CZ16_1_2000m_Temp_year_2018_month_01.nc';

% 读取变量
lat = ncread(file_path, 'lat');
lon = ncread(file_path, 'lon');
time = ncread(file_path, 'time');
depth = ncread(file_path, 'depth_std');
temp = ncread(file_path, 'temp');

% 创建一个表格存储数据
[Lon, Lat, Depth, Time] = ndgrid(lon, lat, depth, time);

% 转置并转换为列向量
Lon = Lon(:);
Lat = Lat(:);
Depth = Depth(:);
Time = Time(:);

% 转置温度数据并转换为列向量
Temp = permute(temp, [2, 3, 1, 4]);
Temp = Temp(:);

% 创建一个表格
T = table(Lon, Lat, Depth, Time, Temp, 'VariableNames', {'Longitude',
'Latitude', 'Depth', 'Time', 'Temperature'});

% 筛选深度在0到60米之间的数据
filtered_T = T(T.Depth <= 60, :);

% 导出到Excel文件
writetable(filtered_T, 'temperature_01.csv');
```

6. SQL Server导入数据建表脚本：

```
CREATE TABLE t_temp_filtered (
    TempID INT IDENTITY(1,1) PRIMARY KEY,
    Lon FLOAT not null,
    Lat FLOAT not null,
    Depth FLOAT not null,
    Time INT not null,
    Temp FLOAT
);

BULK INSERT t_temp_filtered
FROM 'C:\IAP_CZ16_2018\Temperature_01.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);
```

```
BULK INSERT t_temp_filtered
FROM 'C:\IAP_CZ16_2018\Temperature_07.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2
);

-- 删除旧表
DROP TABLE t_temp;

-- 重命名新表为旧表名
EXEC sp_rename 't_temp_filtered', 't_temp';
```

7.4 功能测试

经测试，功能都已实现。

请参见附件`数据库课设演示.mp4`

八.总结

8.1总结

海洋表面温度（Sea Surface Temperature, SST）是气候系统中的关键变量，对全球气候模式和环境变化具有显著影响。本研究旨在设计并开发一个海洋表面温度数据库系统，以支持科研人员、环境保护机构和海洋管理部门对SST数据的有效管理和分析。系统需求分析确定了数据存储、查询、筛选、导出和用户管理等功能需求。采用C#和.NET框架，结合SQL Server数据库管理系统，实现了一个客户端-服务器架构的系统。系统设计遵循模块化原则，确保了扩展性和维护性。通过E-R图转化为关系模型，并进行了数据模型的规范化设计，以提高数据一致性和减少冗余。项目管理采用了WinForms作为前端框架，Entity Framework作为ORM工具，以及Visual Studio作为开发环境。系统实现包括用户登录、权限管理、数据展示、查询、导出等功能，并通过功能测试验证了实现的正确性。本系统为海洋表面温度数据的管理和分析提供了一个高效、准确、易用的解决方案，有助于用户更好地理解和应对气候变化的挑战。

8.2 项目心得

在完成海洋表面温度数据库系统的过程中，我经历了从需求分析、设计、实现到测试和部署的开发周期，积累了一定的经验。以下是一些主要的项目心得：

1. **全面的需求分析**： 在项目初期，通过需求分析，明确了系统的功能需求和期望。这一步骤确保了我在后续设计和开发过程中能够明确目标，减少返工和调整，提高开发效率。
2. **合理的架构设计**： 系统架构的合理设计是项目成功的基础。我选择了C#和.NET框架作为开发语言和平台，利用WinForms构建用户界面，SQL Server作为数据库管理系统，Entity Framework作为数据访问层。这些技术的组合不仅提高了开发效率，还确保了系统的稳定性和可扩展性。
3. **有效的时间管理**： 时间管理显得尤为重要。我制定了项目计划和时间表，将项目分解为多个可管理的小任务，逐一完成。通过进度跟踪和每周的总结，我能够及时发现问题并调整计划，确保项目按时完成。

4. **技术学习与应用**： 在项目开发过程中，我学习并应用了多种新技术和工具，CsvHelper用于CSV文件处理。通过实际项目的应用，我不仅掌握了这些工具的使用方法，还提高了自己的技术能力。
5. **灵活应对问题和挑战**： 项目过程中，我遇到了一些技术难题和挑战，如多条件查询的实现、数据导出的性能优化等。通过查阅文档、在线资料和社区论坛，我找到了有效的解决方案。这些挑战锻炼了我的问题解决能力。
6. **用户体验设计**： 用户体验设计在本项目中占据重要地位。我注重界面的简洁性和操作的便捷性，使得用户能够轻松进行数据查询、筛选和导出操作。
7. **项目管理和自我激励**： 在整个开发过程中，我通过设置短期目标和奖励机制，自我激励，保持工作状态。

通过本次海洋表面温度数据库系统的开发，我大致实现了预期的功能目标，还在软件开发的方面积累了一些的经验。需求分析、架构设计、时间管理、技术学习与应用、问题解决以及项目管理等各个环节的有效执行，为项目的成功交付提供了保障。此次项目的顺利完成，不仅提高了我的技术能力和项目管理能力，也增强了我在未来项目中的热情。

九.参考文献

C# 图书管理系统 winform 入门教程 SqlServer数据库*哔哩哔哩*bilibili

国家海洋科学数据中心 (nmdis.org.cn)

Kimi.ai - 帮你看更大的世界 (moonshot.cn)

ChatGPT