**数据库设计**

**人工智能221 何昂202210310219**

1.对清舱机器人进行实时监控和控制，保障机器人的安全性能和可靠运行。

2.优化清舱机器人运营管理，提高机器人的工作效率和经济效益。

3.实现机器人信息化和智能化，提高企业和用户对机器人的决策和管理水平。

Admin 表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 是否为空 | 说明 |
| id | int | 否 | 自增主键 |
| user\_account | varchar(50) | 否 | 账号 |
| password | varchar(50) | 否 | 密码 |

Goods 表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 是否为空 | 说明 |
| goods\_no | int | 否 | 主键 |
| start\_address | varchar(50) | 否 |  |
| transport\_address | varchar(50) | 否 |  |
| ship\_no | varchar(50) | 否 |  |
| is\_liquid | varchar(5) | 否 |  |
| count | int | 否 |  |
| weight | decimal(9, 2) | 否 |  |

Robot表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 是否为空 | 说明 |
| no | varchar(50) | 否 | 主键 |
| name | varchar(50) | 否 |  |
| password | varchar(50) | 否 |  |
| category | varchar(50) | 否 |  |
| generate\_manufacturer | varchar(50) | 否 |  |
| use\_date | varchar(50) | 否 |  |
| quantity\_electricity | varchar(50) | 否 |  |

Robot\_Plan表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 是否为空 | 说明 |
| plan\_no | varchar(50) | 否 | 主键 |
| robot\_no | varchar(50) | 否 |  |
| length | decimal(9, 2) | 否 |  |
| count | int | 否 |  |
| is\_need\_avoid | varchar(5) | 否 |  |

Ship表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 是否为空 | 说明 |
| ship\_no | varchar(50) | 否 | 主键 |
| type | varchar(50) | 否 |  |
| stop\_wharf | varchar(50) | 否 |  |
| robot\_no | varchar(50) | 否 |  |
| voyage | varchar(50) | 否 |  |
| interval\_time | int | 否 |  |
| length | decimal(9, 2) | 否 |  |
| area | decimal(9, 2) | 否 |  |

2.3系统功能分析

1：使用账号和密码登录到系统。

2：可以对机器人信息的增删改查。

3：可以对机器人路径的增删改查。

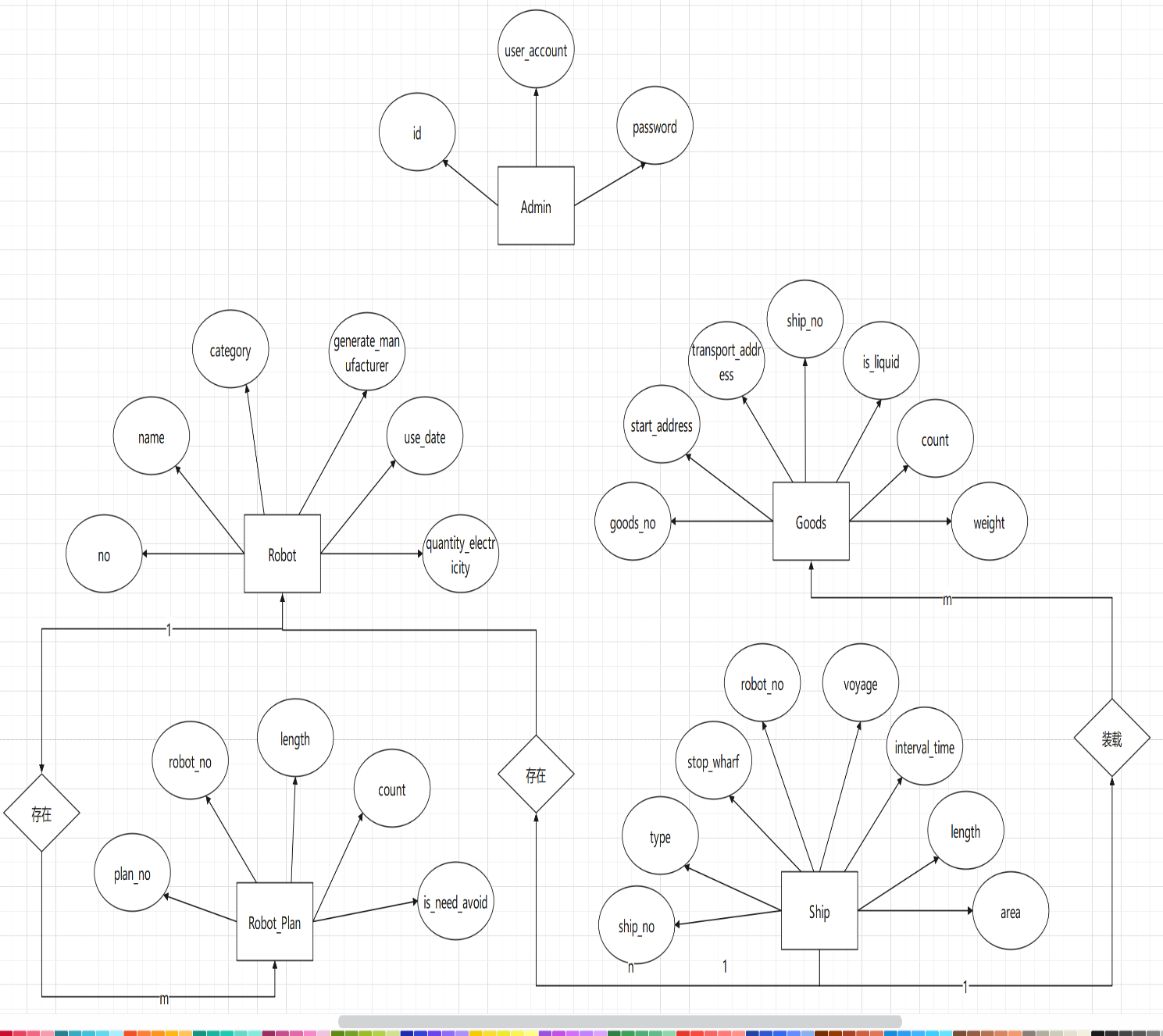
4：可以对散货船的增删改查。

5：可以对货物的增删改查。

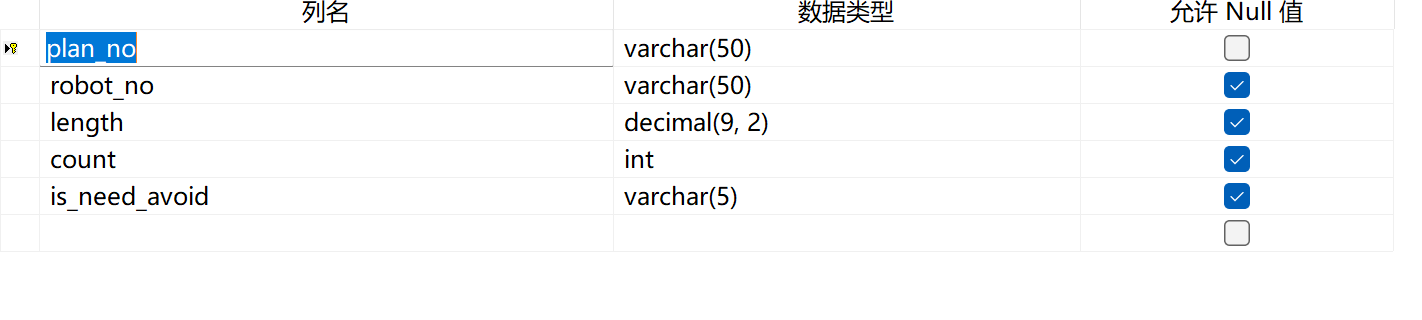
6：统计报表展示各种机器人相关的统计。

2.4概念结构设计

使用er图软件画er图，具体如下图







import pyodbc

import logging

class DatabaseConnection:

    def \_\_init\_\_(self, server, database, username, password):

        self.server = server

        self.database = database

        self.username = username

        self.password = password

        self.connection = None

    def connect(self):

        try:

            self.connection = pyodbc.connect(

                f'DRIVER={{SQL Server}};SERVER={self.server};DATABASE={self.database};UID={self.username};PWD={self.password}'

            )

            logging.info(f"Connected to {self.database} successfully")

        except pyodbc.Error as e:

            logging.error(f"Error connecting to SQL Server: {e}")

    def close(self):

        if self.connection:

            self.connection.close()

            logging.info("Connection closed")

    def create\_tables(self):

        try:

            cursor = self.connection.cursor()

            # 创建用户表

            cursor.execute('''

                IF NOT EXISTS (SELECT \* FROM sysobjects WHERE name='user' AND xtype='U')

                CREATE TABLE user (

                    user\_id INT PRIMARY KEY IDENTITY(1,1),

                    username NVARCHAR(50) UNIQUE NOT NULL,

                    password NVARCHAR(50) NOT NULL

                )

            ''')

            # 创建路径数据表

            cursor.execute('''

                IF NOT EXISTS (SELECT \* FROM sysobjects WHERE name='path\_data' AND xtype='U')

                CREATE TABLE path\_data (

                    path\_id INT PRIMARY KEY IDENTITY(1,1),

                    path\_name NVARCHAR(100) NOT NULL,

                    real\_coordinates NVARCHAR(200) NOT NULL,

                    set\_coordinates NVARCHAR(200) NOT NULL,

                    error FLOAT NOT NULL

                )

            ''')

            self.connection.commit()

            logging.info("Tables created successfully")

        except pyodbc.Error as e:

            logging.error(f"Error creating tables: {e}")

            self.connection.rollback()

    # 插入用户数据

    def insert\_user(self, username, password):

        try:

            cursor = self.connection.cursor()

            cursor.execute(

                "INSERT INTO user (username, password) VALUES (?, ?)",

                (username, password)

            )

            self.connection.commit()

            logging.info("User inserted successfully")

        except pyodbc.Error as e:

            logging.error(f"Error inserting user: {e}")

            self.connection.rollback()

    # 插入路径数据

    def insert\_path\_data(self, path\_name, real\_coordinates, set\_coordinates, error):

        try:

            cursor = self.connection.cursor()

            cursor.execute(

                "INSERT INTO path\_data (path\_name, real\_coordinates, set\_coordinates, error) VALUES (?, ?, ?, ?)",

                (path\_name, real\_coordinates, set\_coordinates, error)

            )

            self.connection.commit()

            logging.info("Path data inserted successfully")

        except pyodbc.Error as e:

            logging.error(f"Error inserting path data: {e}")

            self.connection.rollback()

    # 查询用户

    def query\_user(self, username, password):

        try:

            cursor = self.connection.cursor()

            cursor.execute("SELECT \* FROM user WHERE username=? AND password=?", (username, password))

            user = cursor.fetchone()

            return user

        except pyodbc.Error as e:

            logging.error(f"Error querying user: {e}")

            return None

    # 查询路径数据

    def query\_path\_data(self):

        try:

            cursor = self.connection.cursor()

            cursor.execute("SELECT \* FROM path\_data")

            paths = cursor.fetchall()

            return paths

        except pyodbc.Error as e:

            logging.error(f"Error querying path data: {e}")

            return []

# 用法示例

if \_\_name\_\_ == "\_\_main\_\_":

    db = DatabaseConnection(

        server="localhost",

        database="robot\_system",  # 数据库名称

        username="sa",  # 数据库用户名

        password="yourpassword"  # 数据库密码

    )

    db.connect()

    db.create\_tables()

    # 插入数据示例

    db.insert\_user("admin", "admin123")

    db.insert\_path\_data("TestPath", "[(1,1), (2,2)]", "[(1,1), (2,2)]", 0.1)

    # 查询数据示例

    user = db.query\_user("admin", "admin123")

    print("Queried User:", user)

    paths = db.query\_path\_data()

    print("Queried Path Data:", paths)

    db.close()