**《 智能系统 》实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学号** | | **姓名** | **承担任务** | | | **贡献度** | **得分** |
| **20194053** | | **王志鹏** | **编写上、下位机、数据库代码及实验报告撰写** | | |  |  |
| **20194051** | | **王良厅** | **编写上、下位机、数据库代码** | | |  |  |
| **20184199** | | **王赛宇** | **模糊知识模块设计** | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
|  | |  |  | | |  |  |
| **实验题目** | **数据库与知识库设计** | | | | | | |
| **实验时间** | **2022/4/15** | | | **实验地点** | **DS3401** | | |
| **实验成绩** |  | | | **实验性质** | **□验证性 □设计性 ☑综合性** | | |
| 教师评价：  □算法/实验过程正确 □源程序/实验内容提交  □程序结构/实验步骤合理 □实验结果正确  □语法、语义正确 □报告规范  其他：  评价教师签名： | | | | | | | |
| 1. 实验目的   为实现十字路口红绿灯智能控制，本次实验的目的是：  （1）设计数据库  （2）设计可信度知识与模糊知识  （3）设计知识库  （4）实现知识库 | | | | | | | |
| 1. 实验项目内容   1、设计并实现数据库  （1）设计并建立存储下位机上传数据的表（给出表结构）  （2）将下位机上传的数据存入数据库（给出程序流程）；  （3）设计并建立存储上位机输出数据（信号）的表（给出表结构）；  （4）将上位机输出数据（信号）存入数据库（给出程序流程）。  2、可信度知识设计  （1）设计可信度知识表示；  （2）表示所有可能使用的可信度知识（不少于3条）。  3、模糊知识设计  （1）设计模糊知识表示；  （2）列举所有可能使用的模糊知识（不少于3条）；  （3）给出（2）中所有模糊集的隶属函数；  （4）设计并实现模糊矩阵计算程序（函数）。  4、基于RDBMS设计并实现知识库  （1）设计并实现可信度知识存储表结构，将第2项所列知识存入；  （2）设计并实现模糊知识存储表结构，将第3项所列知识存入；  （3）设计并实现知识维护程序（函数）；  （4）设计并实现知识查询查询（函数）。 | | | | | | | |
| 三、实验过程或算法（代码）   1. 设计并实现数据库： 2. 用户管理表结构：  |  |  | | --- | --- | | 编号 | 姓名 |  1. 车流量管理表结构：  |  |  |  |  | | --- | --- | --- | --- | | 编号 | 方向 | 车流量 | 时间 |  1. 知识管理表结构：  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 编号 | 前提 | 结论 | 前提可信度 | 知识可信度 |  1. 将数据存入数据库：  * 用户数据：   首先建立一个User类：   1. class User(Base): 2. """User Roles""" 3. \_\_tablename\_\_ = 'user' 4. id = Column(Integer, primary\_key=True, autoincrement=True) 5. name = Column(Unicode(255), unique=True) 6. def \_\_repr\_\_(self): 7. return "<{} {} >".format( 8. self.\_\_class\_\_.\_\_name\_\_, 9. self.name, 10. ) 11. @classmethod 12. def find(cls, db, name): 13. """Find a role by name. 14. Returns None if not found. 15. """ 16. return db.query(cls).filter(cls.name == name).first() 17. @classmethod 18. def findById(cls, db, id): 19. """Find a user by name. 20. Returns None if not found. 21. """ 22. return db.query(cls).filter(cls.id == id).first()   随后在dbUtil文件当中添加相应的数据库操作函数：   1. def addUser(db,username): 2. user = orm.User(name=username) 3. db.add(user) 4. db.commit() 5. def updateUser(db,id,username): 6. user = orm.User.findById(db,id) 7. user.name = username 8. db.commit() 9. def deleteUser(db,id): 10. user = orm.User.findById(db,id) 11. db.delete(user) 12. db.commit() 13. def findUserList(db): 14. users = db.query(orm.User).all() 15. return users  * 车流量数据：基本与用户数据一致，代码如下  1. class Car(Base): 2. """User Roles""" 3. \_\_tablename\_\_ = 'car' 4. id = Column(Integer, primary\_key=True, autoincrement=True) 5. direction = Column(Unicode(255), unique=True) 6. flow = Column(Integer) 7. time = Column(Unicode(255), unique=True) 8. def \_\_repr\_\_(self): 9. return "<{} {} {} {}>".format( 10. self.\_\_class\_\_.\_\_name\_\_, 11. self.direction, 12. self.flow, 13. self.time 14. ) 15. @classmethod 16. def findByFlow(cls, db, flow): 17. """Find a role by flow. 18. Returns None if not found. 19. """ 20. return db.query(cls).filter(cls.flow == flow).first() 21. @classmethod 22. def findById(cls, db, id): 23. """Find a user by id. 24. Returns None if not found. 25. """ 26. return db.query(cls).filter(cls.id == id).first() 27. @classmethod 28. def findByDirection(cls, db, direction): 29. """Find a user by direction. 30. Returns None if not found. 31. """ 32. return db.query(cls).filter(cls.direction == direction).first() 33. @classmethod 34. def findByTime(cls, db, time): 35. """Find a user by time. 36. Returns None if not found. 37. """ 38. return db.query(cls).filter(cls.time == time).first() 39. 可信度知识设计：   主要由四部分构成：前提、结论、前提可信度、知识可信度   1. class Knowledge(Base): 2. """Knowledge Roles""" 3. \_\_tablename\_\_ = 'knowledge' 4. *#ID* 5. id = Column(Integer, primary\_key=True, autoincrement=True) 6. *#前提* 7. E = Column(Unicode(255)) 8. *#结论* 9. H = Column(Unicode(255)) 10. *#前提可信度* 11. CFE = Column(Float) 12. *#知识可信度* 13. CFHE = Column(Float) 14. def \_\_repr\_\_(self): 15. return "<{} {} {} {} {}>".format( 16. self.\_\_class\_\_.\_\_name\_\_, 17. self.E, 18. self.H, 19. self.CFE, 20. self.CFHE 21. ) 22. 基于RDBMS设计并实现知识库： 23. 表的建立、知识类的定义： 24. class Knowledge(Base): 25. """Knowledge Roles""" 26. \_\_tablename\_\_ = 'knowledge' 27. *#ID* 28. id = Column(Integer, primary\_key=True, autoincrement=True) 29. *#前提* 30. E = Column(Unicode(255)) 31. *#结论* 32. H = Column(Unicode(255)) 33. *#前提可信度* 34. CFE = Column(Float) 35. *#知识可信度* 36. CFHE = Column(Float) 37. def \_\_repr\_\_(self): 38. return "<{} {} {} {} {}>".format( 39. self.\_\_class\_\_.\_\_name\_\_, 40. self.E, 41. self.H, 42. self.CFE, 43. self.CFHE 44. ) 45. 知识维护程序： 46. *# 可信度知识* 47. def addKL(db,E,H,CFE,CFHE): 48. kl = orm.Knowledge(E=E,H=H,CFE=CFE,CFHE=CFHE) 49. db.add(kl) 50. db.commit() 51. def updateKL(db,id,E,H,CFE,CFHE): 52. kl = orm.Knowledge.findByID(db,id) 53. kl.E = E 54. kl.H = H 55. kl.CFE = CFE 56. kl.CFHE = CFHE 57. db.commit() 58. def deleteKL(db,id): 59. *# rule = orm.Knowledge.findById(db,id)* 60. r = orm.Knowledge.findByID(db,id) 61. db.delete(r) 62. db.commit() 63. def findKLList(db): 64. KLS=db.query(orm.Knowledge).all() 65. return KLS 66. 知识查询函数： 67. @classmethod 68. def findByID(cls, db, id): 69. """Find a role by ID. 70. Returns None if not found. 71. """ 72. return db.query(cls).filter(cls.id == id).first() 73. @classmethod 74. def findByE(cls, db, E): 75. """Find a role by E. 76. Returns None if not found. 77. """ 78. return db.query(cls).filter(cls.E == E).first() 79. @classmethod 80. def findByH(cls, db, H): 81. """Find a user by H. 82. Returns None if not found. 83. """ 84. return db.query(cls).filter(cls.H == H).first() 85. @classmethod 86. def findByCFE(cls, db, CFE): 87. """Find a user by CFE. 88. Returns None if not found. 89. """ 90. return db.query(cls).filter(cls.CFE == CFE).first() 91. @classmethod 92. def findByCFHE(cls, db, CFHE): 93. """Find a user by CFHE. 94. Returns None if not found. 95. """ 96. return db.query(cls).filter(cls.CFHE == CFHE).first() | | | | | | | |
| 1. 实验结果及分析 2. 用户管理界面：      1. 添加用户：      1. 编辑用户：      1. 车流量管理界面：      1. 添加车流量：      1. 知识管理界面：      1. 添加知识：      1. 模糊知识设计   （1）设计模糊知识表示；  表示形式：  IF x is A THEN y is B （CF, λ）;其中CF、λ∈[0,1)  各项含义：  A、B：模糊集。由 A、B 已经获得模糊矩阵 R．  λ：限制知识使用的阈值  CF：知识的可信度因子。  （2）列举所有可能使用的模糊知识（不少于3条）；  定义radix\_rate为东西方向车辆数目num\_WE与车辆总数num之比，radix\_time为东西方向亮绿灯时间占绿灯总时长的比例，则有：  IF radix\_rate较小 THEN radix\_time较短 (1,0.9)  IF radix\_rate小 THEN radix\_time 短 (1,0.7)  IF radix\_rate正常 THEN radix\_time正常 (1,0.9)  IF radix\_rate较大 THEN radix\_time较长 (1,0.9)  IF radix\_rate长 THEN radix\_time 长 (1,0.9)  （3）给出（2）中所有模糊集的隶属函数；   |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | radix\_rate | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 | | 小 | 1 | 0.8 | 0.6 | 0.4 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | | 较小 | 0.2 | 0.5 | 0.8 | 0.6 | 0.5 | 0.2 | 0 | 0 | 0 | 0 | 0 | | 正常 | 0 | 0 | 0 | 0.2 | 0.5 | 1 | 0.5 | 0.2 | 0 | 0 | 0 | | 较多 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.8 | 0.9 | 0.6 | 0.1 | 0 | | 多 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.3 | 0.7 | 0.9 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | radix\_time | 0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 | | 短 | 1 | 0.8 | 0.6 | 0.4 | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | | 较短 | 0.2 | 0.5 | 0.8 | 0.6 | 0.5 | 0.2 | 0 | 0 | 0 | 0 | 0 | | 正常 | 0 | 0 | 0 | 0.2 | 0.5 | 1 | 0.5 | 0.2 | 0 | 0 | 0 | | 较长 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0.8 | 0.9 | 0.6 | 0.1 | 0 | | 长 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.3 | 0.7 | 0.9 | 1 | | | | | | | | |