实验报告

16337341 朱志儒

1. **数据库建立**

本次实验使用python的pyodbc库建立数据库。

1. **定义数据库**

创建名为QQ\_Speed的数据库：

1. **create** **database** QQ\_Speed
2. **创建基本表**

在QQ\_Speed数据库中创建11个基本表，为不同的列选择合适的数据类型，创建列级完整性约束。

1. 创建car表：
2. **create** **table** car(
3. car\_name **varchar**(100) **primary** **key**,
4. car\_level **varchar**(10),
5. car\_charmingvalue **int**,
6. car\_description **varchar**(7000),
7. car\_price **int**
8. );
9. 创建map表：
10. **create** **table** map(
11. map\_name **varchar**(100) **primary** **key**,
12. star\_level **int**,
13. map\_description **varchar**(7000),
14. best\_record **int**
15. );
16. 创建pet表：
17. **create** **table** pet(
18. pet\_name **varchar**(100) **primary** **key**,
19. pet\_charmingvalue **int**,
20. pet\_description **varchar**(7000),
21. pet\_price **int**
22. );
23. 创建clothes表：
24. **create** **table** clothes(
25. clothes\_name **varchar**(100) **primary** **key**,
26. clothes\_charmingvalue **int**,
27. clothes\_price **int**
28. );
29. 创建player表：
30. **create** **table** player(
31. **name** **varchar**(100) **primary** **key**,
32. **password** **varchar**(100),
33. current\_experience **int**,
34. current\_level **int**,
35. current\_charming\_level **int**,
36. player\_level1 **varchar**(20),
37. player\_level2 **int**,
38. player\_level3 **int**,
39. money **int**
40. );
41. 创建fleet表：
42. **create** **table** fleet(
43. fleet\_name **varchar**(100) **primary** **key**,
44. captain **varchar**(100) **references** player(**name**),
45. fleet\_level1 **varchar**(20),
46. fleet\_level2 **int**,
47. fleet\_level3 **int**
48. );
49. 创建cars\_of\_player表：
50. **create** **table** cars\_of\_player(
51. player\_name **varchar**(100) **references** player(**name**),
52. car\_name **varchar**(100) **references** car(car\_name),
53. is\_driving **int**,
54. **primary** **key**(player\_name, car\_name)
55. );
56. 创建pets\_of\_player表：
57. **create** **table** pets\_of\_player(
58. player\_name **varchar**(100) **references** player(**name**),
59. pet\_name **varchar**(100) **references** pet(pet\_name),
60. pet\_level **int**,
61. pet\_experience **int**,
62. is\_taking **int**,
63. **primary** **key**(player\_name, pet\_name)
64. );
65. 创建maps\_of\_player表：
66. **create** **table** maps\_of\_player(
67. player\_name **varchar**(100) **references** player(**name**),
68. map\_name **varchar**(100) **references** map(map\_name),
69. record **int**,
70. **primary** **key**(player\_name, map\_name)
71. );
72. 创建clothes\_of\_player表：
73. **create** **table** clothes\_of\_player(
74. player\_name **varchar**(100) **references** player(**name**),
75. clothes\_name **varchar**(100) **references** clothes(clothes\_name),
76. iswearing **int**,
77. **primary** **key**(player\_name, clothes\_name)
78. );
79. 创建fleet\_of\_player表：
80. **create** **table** fleet\_of\_player(
81. player\_name **varchar**(100) **references** player(**name**) **primary** **key**,
82. fleet\_name **varchar**(100) **references** fleet(fleet\_name),
83. contribution **int**
84. );
85. **触发器设置**

1）在maps\_of\_player表上定义after update触发器，用户更新某个赛道的比赛记录后，自动判断该记录与赛道的最好记录，如果该记录小于最好记录，则将赛道的最好记录更新为该记录。

1. **create** **trigger** TRI\_Record\_Update
2. **on** maps\_of\_player
3. **after** **update**
4. **as**
5. if (**update**(record))
6. **begin**
7. **declare** @new\_record **int**, @current\_best\_record **int**, @new\_map\_name **varchar**(100);
8. **select** @new\_record = record, @new\_map\_name = map\_name
9. **from** inserted;
10. **select** @current\_best\_record = best\_record
11. **from** map
12. **where** map\_name = @new\_map\_name;
13. if (@new\_record < @current\_best\_record)
14. **update** map
15. **set** best\_record = @new\_record
16. **where** map\_name = @new\_map\_name;
17. **end**

2）在maps\_of\_player表上定义after insert触发器，用户插入某个赛道的比赛记录后，自动判断该记录与赛道的最好记录，如果该记录小于最好记录，则将赛道的最好记录更新为该记录。

1. **create** **trigger** TRI\_Record\_Insert
2. **on** maps\_of\_player
3. **after** **insert**
4. **as**
5. **declare** @new\_record **int**, @current\_best\_record **int**, @new\_map\_name **varchar**(100);
6. **select** @new\_record = record, @new\_map\_name = map\_name
7. **from** inserted;
8. **select** @current\_best\_record = best\_record
9. **from** map
10. **where** map\_name = @new\_map\_name;
11. if (@new\_record < @current\_best\_record)
12. **update** map
13. **set** best\_record = @new\_record
14. **where** map\_name = @new\_map\_name;

3）在pets\_of\_palyer表上定义instead of update触发器，当更新用户宠物的经验值时，自动判断是否符合升级条件，如果该经验值大于或等于宠物当前等级 x 100，则宠物将升级，同时将限制宠物的最高等级为30级。

1. **create** **trigger** TRI\_Pets\_Of\_Player\_Update
2. **on** pets\_of\_player
3. **instead** **of** **update**
4. **as**
5. if (**update**(pet\_experience))
6. **begin**
7. **declare** @new\_player\_name **varchar**(100), @new\_pet\_level **int**, @new\_pet\_name **varchar**(100), @new\_pet\_experiece **int**;
8. **select** @new\_player\_name = player\_name, @new\_pet\_level = pet\_level, @new\_pet\_name = pet\_name, @new\_pet\_experiece = pet\_experience
9. **from** inserted;
10. if (@new\_pet\_experiece < @new\_pet\_level \* 100)
11. **update** pets\_of\_player
12. **set** pet\_experience = @new\_pet\_experiece
13. **where** player\_name = @new\_player\_name and pet\_name = @new\_pet\_name;
15. while @new\_pet\_experiece >= @new\_pet\_level \* 100
16. **begin**
17. if (@new\_pet\_level < 30)
18. **update** pets\_of\_player
19. **set** pet\_level += 1, pet\_experience = @new\_pet\_experiece - @new\_pet\_level \* 1000
20. **where** player\_name = @new\_player\_name and pet\_name = @new\_pet\_name;
21. **else**
22. **update** pets\_of\_player
23. **set** pet\_experience = 30 \* 100
24. **where** player\_name = @new\_player\_name and pet\_name = @new\_pet\_name;
25. **select** @new\_pet\_experiece -= @new\_pet\_level \* 1000, @new\_pet\_level += 1;
26. **end**
27. **end**
29. if (**update**(is\_taking))
30. **begin**
31. **declare** @new\_n\_player\_name **varchar**(100), @ntew\_n\_pet\_name **varchar**(100), @new\_taking **int**;
32. **select** @new\_n\_player\_name = player\_name, @ntew\_n\_pet\_name = pet\_name, @new\_taking = is\_taking
33. **from** inserted;
34. **update** pets\_of\_player
35. **set** is\_taking = @new\_taking
36. **where** @new\_n\_player\_name = player\_name and @ntew\_n\_pet\_name = pet\_name;
37. **end**

4）在pets\_of\_palyer表上定义after pets\_of\_player触发器，当用户携带某个宠物时，自动增加该用户的魅力值；当用户不携带某个宠物时，自动减少该用户的魅力值。

1. **create** **trigger** TRI\_Change\_Pet\_Taking
2. **on** pets\_of\_player
3. **after** **update**
4. **as**
5. if (**update**(is\_taking))
6. **begin**
7. **declare** @new\_is\_taking **int**, @new\_pop\_pl\_peopayecpr\_name **varchar**(100), @new\_pett\_name **varchar**(100);
8. **select** @new\_is\_taking = is\_taking, @new\_pop\_pl\_peopayecpr\_name = player\_name, @new\_pett\_name = pet\_name
9. **from** inserted;
10. if (@new\_is\_taking = 1)
11. **update** player
12. **set** current\_charming\_level += (**select** pet\_charmingvalue
13. **from** pet
14. **where** pet\_name = @new\_pett\_name)
15. **where** **name** = @new\_pop\_pl\_peopayecpr\_name;
16. **else**
17. **update** player
18. **set** current\_charming\_level -= (**select** pet\_charmingvalue
19. **from** pet
20. **where** pet\_name = @new\_pett\_name)
21. **where** **name** = @new\_pop\_pl\_peopayecpr\_name;
22. **end**

5）在pets\_of\_player表上定义after insert触发器，用户购买某个宠物后，自动减少该用户的金币。

1. **create** **trigger** TRI\_Pets\_Of\_Player\_Insert
2. **on** pets\_of\_player
3. **after** **insert**
4. **as**
5. **declare** @new\_plpayer\_name **varchar**(100), @new\_pet\_nname **varchar**(100);
6. **select** @new\_plpayer\_name = player\_name, @new\_pet\_nname = pet\_name
7. **from** inserted;
8. **update** player
9. **set** money -= (**select** pet\_price
10. **from** pet
11. **where** pet\_name = @new\_pet\_nname)
12. **where** **name** = @new\_plpayer\_name;

6）在cars\_of\_player表上定义after update触发器，当用户驾驶某辆车时，自动增加该用户的魅力值；当用户不驾驶某辆车时，自动减少该用户的魅力值。

1. **create** **trigger** TRI\_Change\_Car\_Driving
2. **on** cars\_of\_player
3. **after** **update**
4. **as**
5. if (**update**(is\_driving))
6. **begin**
7. **declare** @new\_is\_driving **int**, @newcp\_player\_name **varchar**(100), @new\_car\_name **varchar**(100);
8. **select** @new\_is\_driving = is\_driving, @newcp\_player\_name = player\_name, @new\_car\_name = car\_name
9. **from** inserted;
10. if (@new\_is\_driving = 1)
11. **update** player
12. **set** current\_charming\_level += (**select** car\_charmingvalue
13. **from** car
14. **where** car\_name = @new\_car\_name)
15. **where** **name** = @newcp\_player\_name;
16. **else**
17. **update** player
18. **set** current\_charming\_level -= (**select** car\_charmingvalue
19. **from** car
20. **where** car\_name = @new\_car\_name)
21. **where** **name** = @newcp\_player\_name;
22. **end**

7）在cars\_of\_player表上定义after insert触发器，用户购买某辆车后，自动减少该用户的金币。

1. **create** **trigger** TRI\_Cars\_Of\_Player\_Insert
2. **on** cars\_of\_player
3. **after** **insert**
4. **as**
5. **declare** @new\_player\_name\_cop **varchar**(100), @new\_car\_name **varchar**(100);
6. **select** @new\_player\_name\_cop = player\_name, @new\_car\_name = car\_name
7. **from** inserted;
8. **update** player
9. **set** money -= (**select** car\_price
10. **from** car
11. **where** car\_name = @new\_car\_name)
12. **where** **name** = @new\_player\_name\_cop;

8）在clothes\_of\_player表上定义after update触发器，当用户穿戴某件服装时，自动增加用户的魅力值；当用户脱下某件服装时，自动减少用户的魅力值。

1. **create** **trigger** TRI\_Change\_Clothes\_Wearing
2. **on** clothes\_of\_player
3. **after** **update**
4. **as**
5. if (**update**(iswearing))
6. **begin**
7. **declare** @new\_iswearing **int**, @copnew\_player\_name **varchar**(100), @new\_clothes\_name **varchar**(100);
8. **select** @new\_iswearing = iswearing, @copnew\_player\_name = player\_name, @new\_clothes\_name = clothes\_name
9. **from** inserted;
10. if (@new\_iswearing = 1)
11. **update** player
12. **set** current\_charming\_level += (**select** clothes\_charmingvalue
13. **from** clothes
14. **where** clothes\_name = @new\_clothes\_name)
15. **where** **name** = @copnew\_player\_name;
16. **else**
17. **update** player
18. **set** current\_charming\_level -= (**select** clothes\_charmingvalue
19. **from** clothes
20. **where** clothes\_name = @new\_clothes\_name)
21. **where** **name** = @copnew\_player\_name;
22. **end**

9）在clothes\_of\_player表上定义after insert触发器，当用户购买某件服装时，自动减少该用户地金币。

1. **create** **trigger** TRI\_Clothes\_Of\_Player\_Insert
2. **on** clothes\_of\_player
3. **after** **insert**
4. **as**
5. **declare** @new\_playercop\_name **varchar**(100),@new\_clothes\_name **varchar**(100);
6. **select** @new\_playercop\_name = player\_name, @new\_clothes\_name = clothes\_name
7. **from** inserted;
8. **update** player
9. **set** money -= (**select** clothes\_price
10. **from** clothes
11. **where** clothes\_name = @new\_clothes\_name)
12. **where** **name** = @new\_playercop\_name;

10）在player表上定义instead of player触发器，当用户更新其经验值时，自动判断是否符合更新条件，如果当前经验值大于或等于当前等级 x 1000，则用户将升级；当用户更新其段位积分时，自动判断是否符合升段条件，如果当前段位积分大于50，则用户将提升小段位（从5到1），如果当前小段位为1，则用户将提升大段位（“青铜”、“白银”、“黄金”、“铂金”、“钻石”、“星耀”、“车神”）。

1. **create** **trigger** TRI\_Player\_Update
2. **on** player
3. **instead** **of** **update**
4. **as**
5. **declare** @new\_name **varchar**(100);
6. if (**update**(money))
7. **begin**
8. **declare** @new\_money **int**;
9. **select** @new\_name = **name**, @new\_money = money
10. **from** inserted;
11. **update** player
12. **set** money = @new\_money
13. **where** **name** = @new\_name;
14. **end**
16. if (**update**(current\_experience))
17. **begin**
18. **declare** @new\_experience **int**, @new\_level **int**;
19. **select** @new\_experience = current\_experience, @new\_level = current\_level, @new\_name = **name**
20. **from** inserted;
21. if (@new\_experience < @new\_level \* 1000)
22. **update** player
23. **set** current\_experience = @new\_experience
24. **where** **name** = @new\_name;
25. while @new\_experience >= @new\_level \* 1000
26. **begin**
27. **update** player
28. **set** current\_level += 1, current\_experience = @new\_experience - @new\_level \* 1000
29. **where** **name** = @new\_name;
30. **select** @new\_experience -= @new\_level \* 1000, @new\_level += 1;
31. **end**
32. **end**
34. if (**update**(current\_charming\_level))
35. **begin**
36. **declare** @new\_charming **int**;
37. **select** @new\_charming = current\_charming\_level, @new\_name = **name**
38. **from** inserted;
39. **update** player
40. **set** current\_charming\_level = @new\_charming
41. **where** **name** = @new\_name;
42. **end**
44. if (**update**(player\_level3))
45. **begin**
46. **declare** @new\_level3 **int**, @new\_level2 **int**, @new\_level1 **varchar**(20), @next\_level1 **varchar**(20);
47. **select** @new\_name = **name**, @new\_level3 = player\_level3, @new\_level2 = player\_level2, @new\_level1 = player\_level1
48. **from** inserted;
49. if (@new\_level3 < 50)
50. **update** player
51. **set** player\_level3 = @new\_level3
52. **where** **name** = @new\_name;
53. while @new\_level3 >= 50
54. **begin**
55. **select** @next\_level1 =
56. case @new\_level1
57. **when** '青铜' **then** '白银'
58. **when** '白银' **then** '黄金'
59. **when** '黄金' **then** '铂金'
60. **when** '铂金' **then** '钻石'
61. **when** '钻石' **then** '星耀'
62. **else** '车神'
63. **end**
64. if (@new\_level2 - 1 < 1)
65. if (@new\_level1 != '车神')
66. **begin**
67. **update** player
68. **set** player\_level2 = 5, player\_level3 = @new\_level3 - 50, player\_level1 = @next\_level1
69. **where** **name** = @new\_name;
70. **select** @new\_level2 = 5, @new\_level1 = @next\_level1;
71. **end**
72. **else**
73. **update** player
74. **set** player\_level3 = 50
75. **where** **name** = @new\_name;
76. **else**
77. **begin**
78. **update** player
79. **set** player\_level2 -= 1, player\_level3 = @new\_level3 - 50
80. **where** **name** = @new\_name;
81. **select** @new\_level2 -= 1;
82. **end**
83. **select** @new\_level3 -= 50;
84. **end**
85. **end**

11）在fleet表上定义的instead of update触发器，当更新车队的段位积分时，自动判断是否符合升段条件，如果当前段位积分大于50，则车队将提升小段位（从5到1），如果当前小段位为1，则车队将提升大段位（“青铜”、“白银”、“黄金”、“铂金”、“钻石”、“星耀”、“车神”）。

1. **create** **trigger** TRI\_Fleet\_Update
2. **on** fleet
3. **instead** **of** **update**
4. **as**
5. **declare** @new\_level1 **varchar**(20), @new\_captain **varchar**(100), @next\_level1 **varchar**(20), @new\_name **varchar**(100), @new\_level2 **int**, @new\_level3 **int**;
6. **select** @new\_name = fleet\_name, @new\_captain = captain, @new\_level1 = fleet\_level1, @new\_level2 = fleet\_level2, @new\_level3 = fleet\_level3
7. **from** inserted;
9. if (**update**(captain))
10. **update** fleet
11. **set** captain = @new\_captain
12. **where** fleet\_name = @new\_name;
14. if (**update**(fleet\_level3))
15. **begin**
16. if (@new\_level3 < 50)
17. **update** fleet
18. **set** fleet\_level3 = @new\_level3
19. **where** fleet\_name = @new\_name;
20. while @new\_level3 >= 50
21. **begin**
22. **select** @next\_level1 =
23. case @new\_level1
24. **when** '青铜' **then** '白银'
25. **when** '白银' **then** '黄金'
26. **when** '黄金' **then** '铂金'
27. **when** '铂金' **then** '钻石'
28. **when** '钻石' **then** '星耀'
29. **else** '车神'
30. **end**
31. if (@new\_level2 - 1 < 1)
32. if (@new\_level1 != '车神')
33. **begin**
34. **update** fleet
35. **set** fleet\_level2 = 5, fleet\_level3 = @new\_level3 - 50, fleet\_level1 = @next\_level1
36. **where** fleet\_name = @new\_name;
37. **select** @new\_level2 = 5, @new\_level1 = @next\_level1;
38. **end**
39. **else**
40. **update** fleet
41. **set** fleet\_level3 = 50
42. **where** fleet\_name = @new\_name;
43. **else**
44. **begin**
45. **update** fleet
46. **set** fleet\_level2 -= 1, fleet\_level3 = @new\_level3 - 50
47. **where** fleet\_name = @new\_name;
48. **select** @new\_level2 -= 1;
49. **end**
50. **select** @new\_level3 -= 50;
51. **end**
52. **end**
53. **应用程序功能实现**
54. **验证登录信息**

根据username在player表中查询password，确认输入的密码是否正确。

1. **def** verifyLogin(username, password):
2. cursor.execute("""
3. select password
4. from player
5. where name = ?
6. """, username)
7. **return** password == cursor.fetchone()[0]
8. 添加新用户

在player表中新增用户username。

1. **def** addplayer(username, password):
2. cursor.execute("""
3. insert into player
4. values (?, ?, 0, 0, 0, '青铜', 5, 0, 100000)
5. """, username, password)
6. cnxn.commit()
7. **return** True
8. 获取车辆信息

如果name为None，则查找car表中所有车辆的信息；如果name为车辆名称，则查找该车辆的相关信息；支持模糊查找，即查找名字包含name的所有车辆的信息。

1. **def** getCar(name = None):
2. cars = []
3. **if** name == None:
4. cursor.execute("""
5. select \*
6. from car
7. """)
8. rows = cursor.fetchall()
9. **for** row **in** rows:
10. cars.append({
11. 'name': row.car\_name,
12. 'level': row.car\_level,
13. 'charming': row.car\_charmingvalue,
14. 'description': row.car\_description,
15. 'price': row.car\_price
16. })
17. **else**:
18. cursor.execute("""
19. select \*
20. from car
21. where car\_name like '%{name}%'
22. """.format(name=name))
23. rows = cursor.fetchall()
24. **for** row **in** rows:
25. cars.append({
26. 'name': row.car\_name,
27. 'level': row.car\_level,
28. 'charming': row.car\_charmingvalue,
29. 'description': row.car\_description,
30. 'price': row.car\_price
31. })
32. **return** cars
33. 获取宠物信息

如果name为None，则查找pet表中所有宠物的信息；如果name为宠物的名称，则查找该宠物的相关信息，支持模糊查找，即查找名字包含name的所有宠物信息。

1. **def** getPet(name = None):
2. pets = []
3. **if** name == None:
4. cursor.execute("""
5. select \*
6. from pet
7. """)
8. rows = cursor.fetchall()
9. **for** row **in** rows:
10. pets.append({
11. 'name': row.pet\_name,
12. 'charming': row.pet\_charmingvalue,
13. 'description': row.pet\_description,
14. 'price': row.pet\_price
15. })
16. **else**:
17. cursor.execute("""
18. select \*
19. from pet
20. where pet\_name like '%{name}%'
21. """.format(name=name))
22. rows = cursor.fetchall()
23. **for** row **in** rows:
24. pets.append({
25. 'name': row.pet\_name,
26. 'charming': row.pet\_charmingvalue,
27. 'description': row.pet\_description,
28. 'price': row.pet\_price
29. })
30. **return** pets
31. 获取服装信息

如果name为None，则查找clothes表中所有的服装信息；如果name为服装名称，则查找该服装的相关信息；支持模糊查找，即查找名字包含name的所有服装信息。

1. **def** getCloth(name = None):
2. clothes = []
3. **if** name == None:
4. cursor.execute("""
5. select \*
6. from clothes
7. """)
8. rows = cursor.fetchall()
9. **for** row **in** rows:
10. clothes.append({
11. 'name': row.clothes\_name,
12. 'charming': row.clothes\_charmingvalue,
13. 'price': row.clothes\_price
14. })
15. **else**:
16. cursor.execute("""
17. select \*
18. from clothes
19. where clothes\_name = '%{name}%'
20. """.format(name=name))
21. rows = cursor.fetchall()
22. **for** row **in** rows:
23. clothes.append({
24. 'name': row.clothes\_name,
25. 'charming': row.clothes\_charmingvalue,
26. 'price': row.clothes\_price
27. })
28. **return** clothes
29. 获取地图信息

如果name为None，则查找map表中所有地图的信息；如果name为地图名字，则查找该地图的相关信息；支持模糊查找，即查找名字包含name的所有地图信息。

1. **def** getMap(name = None):
2. maps = []
3. **if** name == None:
4. cursor.execute("""
5. select \*
6. from map
7. """)
8. rows = cursor.fetchall()
9. **for** row **in** rows:
10. maps.append({
11. 'name': row.map\_name,
12. 'star\_level': row.star\_level,
13. 'map\_description': row.map\_description,
14. 'best\_record': row.best\_record
15. })
16. **else**:
17. cursor.execute("""
18. select \*
19. from map
20. where map\_name like '%{name}%'
21. """.format(name=name))
22. rows = cursor.fetchall()
23. **for** row **in** rows:
24. maps.append({
25. 'name': row.map\_name,
26. 'star\_level': row.star\_level,
27. 'map\_description': row.map\_description,
28. 'best\_record': row.best\_record
29. })
30. **return** maps
31. 获取用户信息

查询player表，获取名字为name的用户的所有信息。

1. **def** getUserInfo(name):
2. cursor.execute("""
3. select \*
4. from player
5. where name = ?
6. """, name)
7. row = cursor.fetchone()
8. userinfo = {
9. 'name': name,
10. 'current\_experience': row.current\_experience,
11. 'experience\_to\_upgrade': row.current\_level \* 1000 - row.current\_experience,
12. 'current\_level': row.current\_level,
13. 'current\_charming\_level': row.current\_charming\_level,
14. 'player\_level\_1': row.player\_level1,
15. 'player\_level\_2': row.player\_level2,
16. 'player\_level\_3': row.player\_level3,
17. 'money': row.money
18. }
19. cursor.execute("""
20. select \*
21. from fleet\_of\_player
22. where player\_name = ?
23. """, name)
24. fleet\_name = cursor.fetchone()
25. **if** fleet\_name != None:
26. userinfo['team'] = fleet\_name.fleet\_name
27. **return** userinfo
28. 获取车队信息

查询fleet\_of\_player表、fleet表、player表，获取车队信息以及车队成员的所有信息。

1. **def** getTeam(name):
2. cursor.execute("""
3. select fleet\_name
4. from fleet\_of\_player
5. where player\_name = ?
6. """, name)
7. row = cursor.fetchone()
8. cursor.execute("""
9. select \*
10. from fleet
11. where fleet\_name = ?
12. """, row.fleet\_name)
13. row = cursor.fetchone()
14. properties = {
15. 'fleet\_name': row.fleet\_name,
16. 'captain': row.captain,
17. 'fleet\_level1': row.fleet\_level1,
18. 'fleet\_level2': row.fleet\_level2,
19. 'fleet\_level3': row.fleet\_level3
20. }
21. cursor.execute("""
22. select player\_name
23. from fleet\_of\_player
24. where fleet\_name = ?
25. """, row.fleet\_name)
26. rows = cursor.fetchall()
27. members = []
28. **for** it **in** rows:
29. members.append(getUserInfo(it.player\_name))
30. **for** i **in** range(len(members)):
31. **if** members[i]['name'] == properties['captain'] **and** i != 0:
32. members[i], members[0] = members[0], members[i]
33. **break**
34. **return** {'properties': properties, 'members': members}
35. 获取物品信息

如果type为user-car，则查询用户name当前驾驶的车辆信息以及他所拥有车辆的信息；如果type为user-pet，则查询用户name当前携带的宠物的信息以及他所拥有宠物的信息；如果type为user-cloth，则查询用户name当前穿戴的服装的信息以及他所拥有服装的信息。

1. **def** getItem(name, type):
2. current = None
3. item\_list = []
4. **if** type == 'user-car':
5. cursor.execute("""
6. select \*
7. from cars\_of\_player, car
8. where player\_name = ? and car.car\_name = cars\_of\_player.car\_name
9. """, name)
10. rows = cursor.fetchall()
11. **for** row **in** rows:
12. **if** row.is\_driving == 1:
13. current = {
14. 'name': row.car\_name,
15. 'level': row.car\_level,
16. 'charming': row.car\_charmingvalue,
17. 'description': row.car\_description,
18. 'price': row.car\_price
19. }
20. **else**:
21. item\_list.append({
22. 'name': row.car\_name,
23. 'level': row.car\_level,
24. 'charming': row.car\_charmingvalue,
25. 'description': row.car\_description,
26. 'price': row.car\_price
27. })
28. **elif** type == 'user-pet':
29. cursor.execute("""
30. select \*
31. from pet, pets\_of\_player
32. where player\_name = ? and pet.pet\_name = pets\_of\_player.pet\_name
33. """, name)
34. rows = cursor.fetchall()
35. **for** row **in** rows:
36. **if** row.is\_taking == 1:
37. current = {
38. 'name': row.pet\_name,
39. 'charming': row.pet\_charmingvalue,
40. 'description': row.pet\_description,
41. 'price': row.pet\_price,
42. 'level': row.pet\_level,
43. 'experience': row.pet\_experience
44. }
45. **else**:
46. item\_list.append({
47. 'name': row.pet\_name,
48. 'charming': row.pet\_charmingvalue,
49. 'description': row.pet\_description,
50. 'price': row.pet\_price,
51. 'level': row.pet\_level,
52. 'experience': row.pet\_experience
53. })
54. **elif** type == 'user-cloth':
55. cursor.execute("""
56. select \*
57. from clothes, clothes\_of\_player
58. where player\_name = ? and clothes.clothes\_name = clothes\_of\_player.clothes\_name
59. """, name)
60. rows = cursor.fetchall()
61. **for** row **in** rows:
62. **if** row.iswearing == 1:
63. current = {
64. 'name': row.clothes\_name,
65. 'charming': row.clothes\_charmingvalue,
66. 'price': row.clothes\_price
67. }
68. **else**:
69. item\_list.append({
70. 'name': row.clothes\_name,
71. 'charming': row.clothes\_charmingvalue,
72. 'price': row.clothes\_price
73. })
74. **if** current == None **and** len(item\_list) == 0:
75. **return** {}
76. **elif** current == None:
77. **return** {'item-list': item\_list}
78. **return** {'current': current, 'item-list': item\_list}
79. 删除成员

从fleet\_of\_player表中删除fleet\_name车队的player\_name成员。

1. **def** removePlayer(fleet\_name, player\_name):
2. cursor.execute("""
3. select player\_name
4. from fleet\_of\_player
5. where fleet\_name = ?
6. """, fleet\_name)
7. rows = cursor.fetchall()
8. notin = True
9. **for** row **in** rows:
10. **if** row.player\_name == player\_name:
11. notin = False
12. **if** notin:
13. **return** False
14. cursor.execute("""
15. delete from fleet\_of\_player
16. where fleet\_name = ? and player\_name = ?
17. """, fleet\_name, player\_name)
18. cnxn.commit()
19. **return** True
20. 添加成员

在fleet\_name车队中添加新成员player\_name。

1. **def** FleetaddPlayer(fleet\_name, player\_name):
2. cursor.execute("""
3. select player\_name
4. from fleet\_of\_player
5. where fleet\_name = ?
6. """, fleet\_name)
7. rows = cursor.fetchall()
8. isin = False
9. **for** row **in** rows:
10. **if** row.player\_name == player\_name:
11. isin = True
12. **if** isin:
13. **return** False
14. cursor.execute("""
15. insert into fleet\_of\_player
16. values (?, ?, 0)
17. """, player\_name, fleet\_name)
18. cnxn.commit()
19. **return** True
20. 购买物品

购买类型为type（车辆、宠物、服装）的商品，即在相应的表（cars\_of\_player、pets\_of\_player、clothes\_of\_player）中插入新的元组，插入后将触发相应的触发器以更新用户user的金币。

1. **def** buy(type, name, user):
2. cursor.execute("""
3. select money
4. from player
5. where name = ?
6. """, user)
7. user\_money = cursor.fetchone().money
8. **if** type == 'car':
9. cursor.execute("""
10. select car\_price
11. from car
12. where car\_name = ?
13. """, name)
14. car\_price = cursor.fetchone().car\_price
15. **if** user\_money < car\_price:
16. **return** False
17. cursor.execute("""
18. insert into cars\_of\_player
19. values (?, ?, 0)
20. """, user, name)
21. **elif** type == 'cloth':
22. cursor.execute("""
23. select clothes\_price
24. from clothes
25. where clothes\_name = ?
26. """, name)
27. clothes\_price = cursor.fetchone().clothes\_price
28. **if** user\_money < clothes\_price:
29. **return** False
30. cursor.execute("""
31. insert into clothes\_of\_player
32. values (?, ?, 0)
33. """, user, name)
34. **elif** type == 'pet':
35. cursor.execute("""
36. select pet\_price
37. from pet
38. where pet\_name = ?
39. """, name)
40. pet\_price = cursor.fetchone().pet\_price
41. **if** user\_money < pet\_price:
42. **return** False
43. cursor.execute("""
44. insert into pets\_of\_player
45. values (?, ?, 0, 0, 0)
46. """, user, name)
47. cnxn.commit()
48. **return** True
49. 装备或卸下物品

如果name为None，则表示卸下物品；如果name为物品名字，则表示装备物品；类型type（车辆、宠物、服装）表示装备或卸下物品的类型。

1. **def** dress(type, name, user):
2. **if** type == 'car':
3. is\_driving = 1
4. **if** name == None:
5. is\_driving = 0
6. cursor.execute("""
7. select car\_name
8. from cars\_of\_player
9. where player\_name = ? and is\_driving = 1
10. """, user)
11. name = cursor.fetchone().car\_name
12. cursor.execute("""
13. update cars\_of\_player
14. set is\_driving = ?
15. where player\_name = ? and car\_name = ?
16. """, is\_driving, user, name)
17. **elif** type == 'pet':
18. is\_taking = 1
19. **if** name == None:
20. is\_taking = 0
21. cursor.execute("""
22. select pet\_name
23. from pets\_of\_player
24. where player\_name = ? and is\_taking = 1
25. """, user)
26. name = cursor.fetchone().pet\_name
27. cursor.execute("""
28. update pets\_of\_player
29. set is\_taking = ?
30. where player\_name = ? and pet\_name = ?
31. """, is\_taking, user, name)
32. **elif** type == 'cloth':
33. iswearing = 1
34. **if** name == None:
35. iswearing = 0
36. cursor.execute("""
37. select clothes\_name
38. from clothes\_of\_player
39. where player\_name = ? and iswearing = 1
40. """, user)
41. name = cursor.fetchone().clothes\_name
42. cursor.execute("""
43. update clothes\_of\_player
44. set iswearing = ?
45. where player\_name = ? and clothes\_name = ?
46. """, iswearing, user, name)
47. cnxn.commit()
48. **return** True
49. 添加比赛记录

在maps\_of\_player表中添加或更新元组表示添加新的比赛记录。每次添加比赛记录将根据赛道的星级增加用户的经验值、金币、段位积分、宠物经验值、车队贡献和车队段位积分，同时将触发相应的触发器以更新用户的等级、段位、宠物等级以及车队段位。

1. **def** addRace(lane, user, score):
2. cursor.execute("""
3. select \*
4. from maps\_of\_player
5. where map\_name = ? and player\_name = ?
6. """, lane, user)
7. row = cursor.fetchone()
8. **if** row == None:
9. cursor.execute("""
10. insert into maps\_of\_player
11. values (?, ?, ?)
12. """, user, lane, score)
13. **elif** score < row.record:
14. cursor.execute("""
15. update maps\_of\_player
16. set record = ?
17. where player\_name = ? and map\_name = ?
18. """, score, user, lane)
19. cursor.execute("""
20. select \*
21. from map
22. where map\_name = ?
23. """, lane)
24. map\_star\_level = cursor.fetchone().star\_level
25. money = map\_star\_level \* 500
26. player\_experience = map\_star\_level \* 1000
27. player\_victory = map\_star\_level \* 25
28. fleet\_contribution = map\_star\_level \* 100
29. fleet\_victory = map\_star\_level \* 10
30. pet\_experience = map\_star\_level \* 100
31. cursor.execute("""
32. update player
33. set money += ?, current\_experience += ?, player\_level3 += ?
34. where name = ?
35. """, money, player\_experience, player\_victory, user)
36. cursor.execute("""
37. select \*
38. from fleet\_of\_player
39. where player\_name = ?
40. """, user)
41. row = cursor.fetchone()
42. **if** row != None:
43. fleet\_name = row.fleet\_name
44. cursor.execute("""
45. update fleet\_of\_player
46. set contribution += ?
47. where player\_name = ?
48. """, fleet\_contribution, user)
49. cursor.execute("""
50. update fleet
51. set fleet\_level3 += ?
52. where fleet\_name = ?
53. """, fleet\_victory, fleet\_name)
54. cursor.execute("""
55. select \*
56. from pets\_of\_player
57. where player\_name = ? and is\_taking = 1
58. """, user)
59. row = cursor.fetchone()
60. **if** row != None:
61. pet\_name = row.pet\_name
62. cursor.execute("""
63. update pets\_of\_player
64. set pet\_experience += ?
65. where player\_name = ? and pet\_name = ?
66. """, pet\_experience, user, pet\_name)
67. cnxn.commit()
68. **return** True

**三、实验总结**

通过这次大作业，我掌握了基于ODBC驱动的数据库应用开发方法。在这次大作业中，我使用的是python的pyodbc库，使用游标创建基本表和设计好的触发器，创建基本表时，借鉴了实验一中的经验；在创建触发器时，借鉴了实验四的经验，使用instead of代替before，从而创建before类型的触发器；使用inserted表示在插入完成后存储所插入行的值，使用deleted存储已经更新或删除行的旧值，从而替代referencing new row as语句和referencing old row as语句。根据前端的需要和之前的设计，我实现相应的应用功能以便前端使用，在实现这些功能时，我使用了预备语句，这既便于实现相应的功能，又能防止SQL注入对数据库造成不可挽回的破坏。