《数据库概论》

实验一：用SQL进行数据操作 实验报告

191220008 陈南疃

QQ：924690736

一、实验环境

操作系统：Windows 10 专业版，64 位操作系统，基于 x64 处理器

软件版本：MYSQL 8.0.26.0

二、实验过程

Q1、

题目：有多少物种 species 的描述 description 中含有单词 “this”？查询以如下形式返回：`(speciesCount)`。

SQL语句：

```
SELECT COUNT(id) AS speciesCount
FROM Species
WHERE description LIKE '%this%';
```

查询结果：

	speciesCount
▶	90

关键思路：

利用 LIKE 谓词筛选出属性 description 中含有单词 “this” 的元组。

Q2、

题目：玩家player ‘Cook’ 将与玩家 player ‘Hughes’ 作战。对于两个玩家，显示他们的用户名 username 和他们各自拥有的 Phonemon 的总能量。查询以如下形式返回：`(username, totalPhonemonPower)`。

SQL语句：

```
SELECT Player.username, SUM(Phonemon.power) AS totalPhonemonPower
FROM Phonemon, Player
WHERE Phonemon.player=Player.id AND
      (Player.username='Cook' OR Player.username='Hughes')
GROUP BY Player.username;
```

查询结果：

	username	totalPhonemonPower
▶	Cook	1220
	Hughes	1170

关键思路：

以 Player.username 作为 GROUP BY 分类条件，分别统计玩家 'Cook' 和 'Hughes' 所拥有的 Phonemon 的总能量。

Q3、

题目：每一个队伍 team 有多少名成员 player？按照玩家数量降序列出队伍名称 title 和玩家数量。查询以如下形式返回：`(title, numberOfPlayers)`。

SQL语句：

```
SELECT Team.title, COUNT(Player.id) AS numberOfPlayers
FROM Team, Player
WHERE Team.id=Player.team
GROUP BY Team.id
ORDER BY numberOfPlayers DESC;
```

查询结果：

	title	numberOfPlayers
▶	Mystic	8
	Valor	6
	Instinct	5

关键思路：

以 Team.id 作为 GROUP BY 分类条件，统计各个队伍的成员人数，最后基于 numberOfPlayers 进行 ORDER BY 降序排列。

Q4、

题目：哪些物种 species 具有类型 type 'grass'？查询以如下形式返回：`(idSpecies, title)`。

SQL语句：

```
SELECT Species.id AS idSpecies, Species.title
FROM Species, Type
WHERE (Species.type1=Type.id AND Type.title='grass') OR
      (Species.type2=Type.id AND Type.title='grass');
```

查询结果：

idSpecies	title
1	Bulbasaur
2	Ivysaur
3	Venusaur
43	Oddish
44	Gloom
45	Vileplume
69	Bellsprout
70	Weepinbell
71	Victreebel
102	Exeggcute
103	Exeggutor
114	Tangela

关键思路：

由于每个 species 具有 1~2 个 type，故联接时需通过 Species.type1 和 Species.type2 两者来判断。

Q5、

题目：列出从未购买过食物 food 的玩家 player 。查询以如下形式返回： `(idPlayer, username)` **。**

SQL语句：

```
SELECT Player.id AS idPlayer, Player.username
FROM Player
WHERE Player.id NOT IN (
    SELECT Purchase.player
    FROM Purchase, Item
    WHERE Purchase.item=Item.id AND Item.type='F');
```

查询结果：

idPlayer	username
4	Reid
7	Hughes
8	Bruce
10	Lyons
11	Emily
12	Darthy
15	Huma

关键思路：

先找出购买过食物的玩家的 id，再用 NOT IN 集合谓词即可筛选出从未购买过食物的玩家。

Q6、

题目：游戏中的每个玩家 player 具有特定的等级 level 。以金额降序列出每一特定等级以及该等级的所有玩家在购买上花费的总金额。查询以如下形式返回： `(level, totalAmountSpentByAllPlayersAtLevel)`**。**

SQL语句：

```
SELECT Player.level, SUM(Item.price*Purchase.quantity) AS
totalAmountSpentByAllPlayersAtLevel
FROM Item, Purchase, Player
WHERE Item.id=Purchase.item AND Purchase.player=Player.id
GROUP BY Player.level
ORDER BY totalAmountSpentByAllPlayersAtLevel DESC;
```

查询结果：

level	totalAmountSpentByAllPlayersAtLevel
2	130.68
12	95.45
6	62.37
5	52.98
3	51.75
1	39.58
4	33.74
8	29.48
11	26.97
7	24.26
10	17.22
9	9.99

关键思路：

将 Item、Purchase、Player 三张表联接后，按照 Player.level 分类，统计各个 level 玩家购买的总金额，最后按金额降序排列。

Q7、

题目：什么物品 item 被购买次数最多？如有并列，列出所有购买次数最多的物品。查询以如下形式返回： `(item, title, numTimesPurchased)`**。**

SQL语句：

```
SELECT Item.id AS item, Item.title, COUNT(*) AS numTimesPurchased
FROM Purchase, Item
WHERE Purchase.item=Item.id
GROUP BY Purchase.item
HAVING COUNT(*)>=ALL (
    SELECT COUNT(*)
    FROM Purchase
    GROUP BY Purchase.item);
```

查询结果：

	item	title	numTimesPurchased
▶	1	Phoneball	10

关键思路：

先按照 Purchase.item 分类，统计出各个物品被购买的次数，找出其中购买次数大于等于每个物品的购买次数的物品即为目标物品。

Q8、

题目：找到可获取的食物的数量，和购买所有种类食物至少各一次的玩家。查询以如下形式返回：`(playerID, username, numberDistinctFoodItemsPurchased)`。

SQL语句：

```
SELECT Player.id AS playerID, Player.username,
       (SELECT COUNT(DISTINCT Item.id)
        FROM Item, Purchase
        WHERE Item.id=Purchase.item AND Item.type='F') AS
numberDistinctFoodItemsPurchased
FROM Player
WHERE NOT EXISTS (
    SELECT *
    FROM (SELECT DISTINCT Item.id
          FROM Item, Purchase
          WHERE Item.id=Purchase.item AND Item.type='F') Allfood
    WHERE NOT EXISTS (
        SELECT *
        FROM Purchase
        WHERE Purchase.player=Player.id AND Purchase.item=Allfood.id));
```

查询结果：

	playerID	username	numberDistinctFoodItemsPurchased
▶	20	Zihan	6

关键思路：

先求出所有的食物作为临时表 Allfood，然后利用除法的在SQL中的实现：不存在没有被目标玩家购买过的食物（即用每个玩家所购买过的食物除以 Allfood 临时表），即可得到目标玩家。

Q9、

题目：将距离最近的两个 Phonemon 之间的欧氏距离称为 X 。计算相互之间距离为 X 的 Phonemon 对的数量。查询以如下形式返回：`(numberOfPhonemonPairs,distanceX)`。

SQL语句：

```
SELECT COUNT(*) AS numberOfPhonemonPairs, ROUND(SQRT(POWER(Pm1.latitude-
Pm2.latitude,2)+POWER(Pm1.longitude-Pm2.longitude,2))*100, 2) AS distanceX
FROM Phonemon Pm1, Phonemon Pm2
WHERE Pm1.id < Pm2.id AND
      ROUND(SQRT(POWER(Pm1.latitude-Pm2.latitude,2)+POWER(Pm1.longitude-
Pm2.longitude,2))*100,2)<=ALL (
          SELECT ROUND(SQRT(POWER(Pm3.latitude-
Pm4.latitude,2)+POWER(Pm3.longitude-Pm4.longitude,2))*100,2)
          FROM Phonemon Pm3, Phonemon Pm4
          WHERE Pm3.id < Pm4.id);
```

查询结果：

	numberOfPhonemonPairs	distanceX
▶	98	0.19

关键思路：

将任意两个 Phonemon 之间的欧氏距离计算出来（利用 Pm1.id < Pm2.id 的方式避免重复计算），若该欧氏距离小于等于所有 Phonemon 之间的欧氏距离，则这两个 Phonemon 满足要求。

Q10、

题目：一些玩家 player 热衷于某种特定类型 type 的 Phonemon 。列出捕捉了任一特定类型 type 中每一物种 species 至少各一个 Phonemon 的玩家的名称以及该类型的名称。查询以如下形式返回：`(username, typeTitle)`。

SQL语句：

```
SELECT Player.username, Type.title AS typeTitle
FROM Player,
     Type,
     (SELECT Type.id AS type, SUM(Species.id) AS speciesSum
      FROM Type, Species
      WHERE Type.id=Species.type1 OR Type.id=Species.type2
      GROUP BY Type.id) typeSpeciesSum,
     (SELECT playerTypeSpecies.player, playerTypeSpecies.type,
SUM(playerTypeSpecies.species) AS speciesSum
      FROM
          ((SELECT Phonemon.player, Species.type1 AS type, Species.id AS species
            FROM Phonemon, Species
            WHERE Phonemon.species=Species.id AND Phonemon.player IS NOT NULL)
            UNION
           (SELECT Phonemon.player, Species.type2 AS type, Species.id AS species
            FROM Phonemon, Species
```

```
            WHERE Phonemon.species=Species.id AND Phonemon.player IS NOT NULL
AND Species.type2 IS NOT NULL)) playerTypeSpecies
      GROUP BY playerTypeSpecies.player, playerTypeSpecies.type)
playerTypeSpeciesSum
WHERE typeSpeciesSum.type=playerTypeSpeciesSum.type AND
      typeSpeciesSum.speciesSum=playerTypeSpeciesSum.speciesSum AND
      Player.id=playerTypeSpeciesSum.player AND
      Type.id=typeSpeciesSum.type;
```

查询结果：

	username	typeTitle
▶	Lyons	Bug
	Lyons	Fairy

关键思路：

该题的关键在于如何判断某玩家是否拥有某一类型 type 中所有物种 species 的 Phonemon，而我将此问题转化为判断某玩家拥有的某 type 中不同 species 的 id 之和，是否等于该 type 中所有 species 的 id 之和。

因此我创建了两张临时表：

第一张表是 typeSpeciesSum，该表中记录了每个 type 中所有 species 的 id 之和，具体如下：

	type	speciesSum
▶	1	1499
	2	589
	3	1291
	4	1487
	5	973
	6	1243
	7	21
	8	279
	9	163
	10	770
	11	2964
	12	667
	13	820
	14	1295
	15	577
	16	444
	17	52
	18	35

第二张表是 playerTypeSpeciesSum，该表中记录了每个 player 拥有的 Phonemon 对应的每个 type 中所有 species 的 id 之和，具体如下（部分）：

player	type	speciesSum
1	7	10
1	14	96
2	12	69
2	1	84
2	5	50
2	7	10
2	4	69
2	3	84
3	5	50
3	7	10
3	12	69
3	4	69
4	1	133
4	7	10
4	12	69
4	4	69

然后将两张临时表做自然联接即可得到拥有某 type 中所有 species 的玩家。

三、实验中遇到的困难及解决办法

1、课件中许多知识点在写代码时才发现自己并不熟悉，且理解不深刻或有误（如：GROUP BY，除法的实现）。通过将这些不熟悉的知识点用代码重现，编写一些简单的查询，并观察查询结果后，加深了对知识点的理解。

2、在除法的实现中用到了临时表，发现临时表的结果无法传到外层查询，具体如下：

```
SELECT ···COUNT(Allfood) AS numberDistinctFoodItemsPurchased
FROM ···
WHERE NOT EXISTS (
      SELECT *
      FROM (SELECT DISTINCT Item.id
            FROM Item, Purchase
            WHERE Item.id=Purchase.item AND Item.type='F') Allfood
      WHERE NOT EXISTS (
            SELECT ···
            FROM ···
            WHERE ···));
```

最初我这样写，但发现运行时会报错，说找不到 Allfood：

Error Code: 1054. Unknown column 'Allfood' in 'field list'

于是我在外层查询里再实现一遍相同的临时表（暂未想到更好的方法），便可正常运行：

```
SELECT ···(SELECT COUNT(DISTINCT Item.id)
           FROM Item, Purchase
           WHERE Item.id=Purchase.item AND Item.type='F') AS
numberDistinctFoodItemsPurchased
FROM ···
WHERE NOT EXISTS (
      SELECT *
      FROM (SELECT DISTINCT Item.id
            FROM Item, Purchase
            WHERE Item.id=Purchase.item AND Item.type='F') Allfood
      WHERE NOT EXISTS (
            SELECT ···
            FROM ···
            WHERE ···));
```

3、在较为复杂的查询问题中（如 Q10），往往需要用到多个子查询或临时表，如果直接编写查询代码可能会理不清里面的逻辑关系。此时，可以将问题拆分为多个子问题，将每个子查询或者将与答案无关但可以辅助自己思考的查询单独打印出来，最后进行子问题的拼接即可。以 Q10 为例：

首先，我进行了一个与解答无关的查询：

```
SELECT Player.username AS username, Type.title AS typeTitle, Species.id AS
speciesID
FROM Phonemon, Species, Player, Type
WHERE Phonemon.species=Species.id AND
      Phonemon.player=Player.id AND
      (Species.type1=Type.id OR Species.type2=Type.id)
ORDER BY username, typeTitle, speciesID;
```

	username	typeTitle	speciesID
▶	Alacia	Grass	69
	Alacia	Poison	69
	Alacia	Psychic	96
	Alice	Normal	133
	Barton	Bug	10
	Barton	Psychic	96
	Bobby	Bug	10
	Bobby	Poison	23
	Bruce	Bug	10
	Bruce	Bug	10
	Bruce	Psychic	96
	Bruce	Psychic	96
	Bruce	Psychic	96

该表记录了每个玩家拥有的 Phonemon 所对应的 speciesID 和 typeTitle。对该表进行了简单的浏览后，发现满足题目要求的玩家特别少，这为我今后的解答提供了检验的标准。

然后，将 Q10 中提到的两张临时表均单独编写并打印出来：

```
SELECT Type.id AS type, COUNT(Species.id) AS speciesSum
FROM Type, Species
WHERE Type.id=Species.type1 OR Type.id=Species.type2
GROUP BY Type.id;
```

	type	speciesSum
▶	1	1499
	2	589
	3	1291
	4	1487
	5	973
	6	1243
	7	21
	8	279
	9	163
	10	770
	11	2964
	12	667
	13	820
	14	1295
	15	577
	16	444
	17	52
	18	35

```
SELECT s.player, s.type, SUM(s.species) AS speciesSum
FROM
    ((SELECT Phonemon.player, Species.type1 AS type, Species.id AS species
      FROM Phonemon, Species
      WHERE Phonemon.species=Species.id AND Phonemon.player IS NOT NULL)
      UNION
     (SELECT Phonemon.player, Species.type2 AS type, Species.id AS species
      FROM Phonemon, Species
      WHERE Phonemon.species=Species.id AND Phonemon.player IS NOT NULL AND
Species.type2 IS NOT NULL)) s
GROUP BY s.player, s.type;
```

	player	type	speciesSum
▶	1	7	10
	1	14	96
	2	12	69
	2	1	84
	2	5	50
	2	7	10
	2	4	69
	2	3	84
	3	5	50
	3	7	10
	3	12	69
	3	4	69
	4	1	133
	4	7	10
	4	12	69
	4	4	69

将两张临时表用简单的变量替换（分别为 t1 和 t2）后，最终的查询就变得十分的简洁：

```
SELECT Player.username, Type.title AS typeTitle
FROM Player, Type, t1, t2
WHERE t1.type=t2.type AND
      t1.speciesSum=t2.speciesSum AND
      Player.id=t2.player AND
      Type.id=t1.type;
```

可见这种方法有利于思路的梳理和 bug 的寻找，最后再将临时表替换进去便可完成查询。

四、参考文献及致谢

仅参考了课件 PPT。