

一. 实验目的

- 1. 认识NULL值在数据库中的特殊含义。
- 2. 了解SQL对空值和空集的处理。
- 3. 熟练使用SQL语句进行对空置、空集相关的操作。

二. 实验内容

通过实验验证 SQL SERVER对NULL的处理，包括：

- 在查询的目标表达式中包含空值的运算。
- 在查询条件中空值与比较运算符的运算结果。
- 使用“ IS NULL”或“ IS NOT NULI”来判断元组该列是否为空值。
- 对存在取空值的列按值进行 ORDER BY排序。
- 使用保留字 DISTINCT对空值的处理。
- 使用 GROUP BY对存在取空值的属性值进行分组。
- 结合分组考察空值对各个集合函数的影响，特别注意对 COUNT(*)和 COUNT(列名)的不同影响。
- 考察结果集是空集时，各个集函数的处理情况。
- 验证嵌套查询中返回空集的情况下与各个谓词的运算结果。
- 进行与空值有关的等值连接运算。

三. 实验结果

- 1. 通过查询选修课程C++的学生的人数，其中成绩合格的学生人数，不合格的学生人数，讨论NULL值的特殊含义

首先查询一下选修C++的学生总人数

```
SELECT COUNT(*) AS '总人数'
FROM CHOICES
WHERE cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)
```

结果消息

	总人数
1	6031

查询已成功执行。(local)\CHONOR (10.0 RTM)DESKTOP-CHONOR\Chonor ...School00:00:001行


查询及格人数

```

SELECT COUNT(*) AS '及格人数'
FROM CHOICES
WHERE score >= 60 AND cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)

```

结果		消息
	及格人数	
1	4817	

 查询已成功执行。
 (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 1 行


查询不及格人数

```

SELECT COUNT(*) AS '不及格人数'
FROM CHOICES
WHERE score < 60 AND cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)

```

结果		消息
	不及格人数	
1	724	

 查询已成功执行。
 (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 1 行

我们可以发现 总人数为6031，及格人数为4817，不及格人数为724，此时 $6031 - 4817 - 724 = 490$ 。那么此时还有490个人为NULL。


查询为NULL人数

```

SELECT COUNT(*) AS 'NULL人数'
FROM CHOICES
WHERE score is null AND cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)

```

结果		消息
	NULL人数	
1	490	

 查询已成功执行。
 (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 1 行

此处的成绩为NULL说明这些学生没有成绩，可能是没有考试之类的。

2. 查询选修课程C++的学生的编号和成绩，使用 ORDER BY按成绩进行排序时,取NULL的项是否出现在结果中？如果有，在什么位置？

```
SELECT sid,score
FROM CHOICES
WHERE cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)
ORDER BY score
```

	sid	score
1	845947855	NULL
2	890918686	NULL
3	898137922	NULL
4	867715893	NULL
5	872519782	NULL

查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 6031 行

将成绩按从小到大排列，NULL值并没有被忽略，而是被当作最小值处理，排在序列的最前面。

3. 在上面的查询的过程中，如果加上保留字 DISTINCT会有什么效果呢？

```
SELECT DISTINCT sid,score
FROM CHOICES
WHERE cid=(
    SELECT cid
    FROM COURSES
    WHERE cname = 'c++'
)
ORDER BY score
```

	sid	score
1	800554079	NULL
2	800575453	NULL
3	800579690	NULL
4	800758208	NULL
5	800895421	NULL

查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 6024 行

DISTINCT同时作用于sid和score，因为sid不同所以不会出现出现所有的NULL合并。

此时发现少了几行，说明有人重考了而且还考了一样的分数。

4. 按年级对所有学生进行分组，能得到多少个组？与现实的情况有什么不同

```
SELECT grade
FROM STUDENTS
GROUP BY grade
ORDER BY grade
```

结果		消息
	grade	
1	NULL	
2	1991	
3	1992	
4	1993	
5	1994	

查询已成功执行。 | (local)\CHONOR (10.0 RTM) | DESKTOP-CHONOR\Chonor ... | School | 00:00:00 | 15 行

此处分成了15个组，此时查询结果存在取值为NULL的项，分组时，取NULL值的会被当作一个分组。但是现实中是不存在NULL的年级。

- 结合分组,使用集合函数求每个课程选修的学生的平均分,总的选课记录数,最高成绩,最低成绩,讨论考察取空值的项对集合函数的作用的影响。

```
SELECT cid,
AVG(score) AS '平均分',
COUNT(*) AS '选课总数',
MAX(score) AS '最高分',
MIN(score) AS '最低分'
FROM CHOICES
GROUP BY cid
ORDER BY cid
```

结果

消息

	cid	平均分	选课总数	最高分	最低分
1	10001	75	5898	99	50
2	10002	75	6013	99	50
3	10003	75	5975	99	50
4	10004	76	6110	99	50
5	10005	76	6031	99	50

查询已成功执行。

(local)\CHONOR (10.0 RTM)

DESKTOP-CHONOR\Chonor ...

School

00:00:00

50 行

集合函数中，除了COUNT(*)计算元组时要把取空值的项计算进去，其他的集合函数(MAX,MIN)都忽略了取空值的项。

- 采用嵌套查询的方式,利用比较运算符和谓词ALL的结合来查询表 STUDENTS中最晚入学的学生年级。当存在 GRADE取空值的项时,考虑可能出现的情况,并解释

```
SELECT DISTINCT grade
FROM STUDENTS
WHERE grade >= ALL(
    SELECT grade
    FROM STUDENTS
)
```

Results window showing a single row of data:

Column 1
grade

Status bar: 查询已成功执行。 (local)\CHONOR (10.0 RTM) DESKTOP-CHONOR\Chonor ... School 00:00:00 0 行

此时因为有NULL，NULL不参与任何比较运算，所以没有一个grade能>=all,所以我们查询不到任何信息。如果我们想要查询到信息，那么我们需要排除NULL的影响。

排除NULL后:

```
SELECT DISTINCT grade
FROM STUDENTS
WHERE grade >= ALL(
    SELECT grade
    FROM STUDENTS
    WHERE grade IS NOT NULL
)
```

结果		消息
	grade	
1	2004	

此时我们可以查询到需要的结果

四. 实验感想

本次实验主要就是要理解数据库中NULL的影响，以及NULL对一些函数，谓词的影响。NULL对于我们查询结果影响导致很多时候我们在查询中需要排除NULL值才能获得正确的查询结果。这次主要就是学习几种判断和处理NULL影响的方法。