SQL简介

演习

3.1使用university schema在SQL中编写以下查询。 （我们建议您实际在数据库上运行这些查询，使用我们在本书网站上提供的示例数据db-book.com。提供了有关设置数据库和加载示例数据的说明。以上网站。）

一个。在Comp中找到课程的标题。科学。有3个学分的系。

湾找到由爱因斯坦教练教授的所有学生的身份证;确保结果中没有重复项。

C。找到任何教练的最高薪水。

d。找到所有获得最高薪水的教师（可能有不止一个薪水相同的教师）。

即查找2009年秋季提供的每个部分的注册情况。

F。在2009年秋季查找所有部分的最大注册人数。

G。查找2009年秋季入学人数最多的部分。

回答：

一个。在Comp中找到课程的标题。科学。有3个学分的系。

选择标题

从课程

其中dept name ='Comp。科学“。

和学分= 3

五

6第3章SQL简介

湾找到由爱因斯坦教练教授的所有学生的身份证;确保结果中没有重复项。

可以通过几种不同的方式回答此查询。一种方法如下。

选择不同的student.ID

from（学生加入使用（ID））

join（讲师加入教使用（ID））使用（课程id，sec id，semester，year）

其中instructor.name ='爱因斯坦'

作为连接的替代方法，使用上面的语法可以通过枚举from子句中的关系，并将ID，课程id，section id，semester和year上的相应连接谓词添加到where子句来编写。

请注意，使用自然连接代替join .. using会导致学生ID与教师ID等同，这是不正确的。

C。找到任何教练的最高薪水。

从教师中选择最大（工资）

d。找到所有获得最高薪水的教师（可能有不止一个薪水相同的教师）。

选择ID，名称

从讲师

薪水=（从教师那里选择最高（薪水））

即查找2009年秋季提供的每个部分的注册情况。编写查询的一种方法如下。

选择课程id，sec id，count（ID）来自部分自然连接需要在哪里学习='秋季'

和年份= 2009年

按课程ID分组，sec id

请注意，如果某个部分没有任何学生参加，则不会出现在结果中。确保此类节显示为0的一种方法是用自然左外连接操作替换自然连接，稍后将在第4章中介绍。另一种方法是在select子句中使用子查询，如下所示。

练习7

选择课程ID，sec id，

（从中选择计数（ID）

其中takes.year = section.year

和takes.semester = section.semester和takes.course id = section.course id and takes.section id = section.section id）

从学期='秋季'和年= 2009年的部分

请注意，如果子查询的结果为空，则聚合函数计数返回值0。

F。在2009年秋季查找所有部分的最大注册量。编写此查询的一种方法如下：

选择最大（注册）

from（选择计数（ID）作为自然连接部分的注册需要在学期='秋季'和年= 2009年

按课程ID分组，sec id）

作为在from子句中使用嵌套子查询的替代方法，可以使用with子句，如本问题下一部分的答案所示。

上述查询中的一个微妙问题是，如果没有任何部分有任何注册，答案将为空，而不是0.我们可以使用子查询，从本问题的前一部分，以确保计数为0在这种情况下。

G。查找2009年秋季中注册次数最多的部分。以下答案使用with子句创建临时视图，从而简化查询。

以sec登记为（

选择课程ID，秒id，计数（ID）作为注册

来自部分自然连接需要学期='秋季'和年= 2009年

按课程ID分组，sec id）

选择课程ID，秒id

来自sec注册

注册=（从第二次注册中选择最大（注册））

也可以在没有with子句的情况下编写查询，但查找注册的子查询将在查询中重复两次。

8第3章SQL简介

3.2假设你有一个关系等级分（等级，分数），它提供

从与数字分数相关的字母等级转换;例如，“A”等级可以被指定为对应于4个点，“A-”对应于3.7个点，“B +”对应于3.3个点，“B”对应于3个点，等等。学生为课程（部分）获得的成绩分数被定义为课程的学分数乘以学生所获得的成绩的数字分数。

鉴于上述关系和我们的大学架构，请在SQL中编写以下每个查询。为简单起见，您可以假设没有使元组具有等级的空值。

一个。在学生所有课程中查找ID为12345的学生所获得的总成绩。

湾找出上述学生的平均成绩（GPA），即总成绩点数除以相关课程的总学分。

C。找到每位学生的身份证和平均分。

回答：

一个。在学生所有课程中查找ID为12345的学生所获得的总成绩。

选择总和（积分\*积分）

从（采取自然联合课程）自然联合成绩点

whereID ='12345'

上述查询的一个问题是，如果学生没有采取任何课程，结果将没有任何元组，而我们希望得到0作为答案。解决这个问题的一种方法是使用自然的左外连接操作，我们将在后面的第4章中讨论。另一种确保我们得到0作为答案的方法是对以下查询：

（选择总和（积分\*积分）

从（采取自然联合课程）自然联合成绩点

其中ID ='12345'）

联盟

（选择0

从学生

其中takes.ID ='12345'和

不存在（select \* from take where takes.ID ='12345'））

像往常一样，可以在where子句中指定连接条件，而不是使用自然连接操作或连接..使用操作。

练习9

湾找出上述学生的平均成绩（GPA），即总成绩点数除以相关课程的总学分。

选择总和（积分\*积分）/总和（积分）作为GPA

从（采取自然联合课程）自然联合成绩点

其中ID ='12345'

和以前一样，没有参加过任何课程的学生不会出现在上述结果中;我们可以通过使用此问题前一部分中的修改后的查询来确保此类学生出现在结果中。但是，在这种情况下的另一个问题是，信用额也将为0，从而导致除以零的条件。实际上，在这种情况下定义GPA的唯一有意义的方法是将其定义为null。通过在上面的查询中添加以下union子句，我们可以确保这样的学生在GPA为null的结果中出现。

联盟

（选择null作为GPA

从学生

其中takes.ID ='12345'和

不存在（select \* from take where takes.ID ='12345'））

其他确保上述方法的方法将在后面的练习4.5的解决方案中讨论。

C。找到每位学生的身份证和平均分。

选择ID，总和（积分\*积分）/总和（积分）作为GPA从（采用自然联接课程）自然联合成绩点组按ID

再次，为了处理没有参加过任何课程的学生，我们必须添加以下的union子句：

联盟

（选择ID，null为GPA

从学生

哪里不存在（select \* from take where takes.ID = student.ID））

3.3

3.4使用大学模式在SQL中编写以下插入，删除或更新。

一个。增加Comp中每位教师的工资。科学。部门减少10％。

湾删除所有从未提供过的课程（即不在章节关系中出现）。

10第3章SQL简介

C。插入每个学生，其cred信息属性大于100，作为同一部门的讲师，薪水为10,000美元。

回答：

一个。增加Comp中每位教师的工资。科学。部门减少10％。

更新讲师

设定工资=工资\* 1.10

其中dept name ='Comp。科学“。

湾删除所有从未提供过的课程（即不在章节关系中出现）。

从课程中删除

其中课程不在

（从部分选择课程ID）

C。插入每个学生，其cred信息属性大于100，作为同一部门的讲师，薪水为10,000美元。

插入讲师

选择ID，名称，部门名称，10000

从学生

其中，信用卡> 100

3.5考虑图??的保险数据库，其中主键带下划线。为此关系数据库构造以下SQL查询。

一个。查找1989年拥有涉及事故的汽车的总人数。

湾向数据库添加新事故;假设所需属性的任何值。

C。删除属于“John Smith”的马自达。

答：注意：参与的关系涉及司机，汽车和事故。

一个。查找1989年拥有涉及事故的汽车的总人数。

注意：这与1989年的事故总数不同。我们必须只计算一次事故的人数。

选择计数（不同的名称）

从事故，参加，人

其中accident.report number = participant.report number

和visited.driver id = person.driver id

日期'1989-00-00'和日期'1989-12-31'之间的日期

练习11

人（驾驶员身份证，姓名，地址）车（执照，型号，年份）

事故（报告编号，日期，地点）拥有（驾驶员身份证，执照）

参加（驾驶员ID，车辆，报告编号，损坏金额）

图??保险数据库。

湾向数据库添加新事故;假设所需属性的任何值。

我们假设司机是“琼斯”，虽然它可能是其他人。此外，我们假设“琼斯”拥有一辆丰田汽车。首先，我们必须找到给定汽车的许可证。然后必须更新参与和事故关系，以便记录事故并将其绑定到给定的汽车。我们假设位置为“Berkeley”，日期和日期为“2001-09-01”，报告编号为4007，损伤金额为3000。

插入事故

价值观（4007，'2001-09-01'，'伯克利'）

插入参加

选择o.driver id，c.license，4007,3000

来自人p，拥有o，汽车c

其中p.name ='琼斯'和p.driver id = o.driver id和

o.license = c.license和c.model ='丰田'

C。删除属于“John Smith”的马自达。

由于模型不是汽车关系的关键，我们可以假设只有约翰史密斯的汽车中的一辆是马自达，或者删除所有的约翰史密斯的马兹达（查询是相同的）。再假设名字是人的关键。

删除汽车

其中model ='Mazda'和许可证

（选择许可证

来自人p，拥有o

其中p.name ='John Smith'和p.driver id = o.driver id）

注意：与马自达相关的拥有，事故和参与记录仍然存在。

3.6假设我们有一个关系标记（ID，得分），我们希望根据得分给学生分配如下：如果得分<40则为F级，如果40≤得分<60则为C级，如果60≤得分则为B级<80，如果80≤分，则为A级。编写SQL查询以执行以下操作：

一个。根据标记关系显示每个学生的成绩。

12第3章SQL简介

湾查找每个年级的学生人数。

回答：

一个。根据标记关系显示每个学生的成绩。

选择ID，

案件

当得分<40时，当得分<60则为'F'，然后当得分<80时为'C'，然后为'B'，否则为'A'

从商标结束

湾查找每个年级的学生人数。

成绩为

（

选择ID，

案件

当得分<40时，当得分<60则为'F'，然后当得分<80时为'C'，然后为'B'，否则为'A'

结束等级

从标记

）

从成绩中选择成绩，计数（ID）

按年级分组

作为替代方案，可以删除with子句，而可以将成绩的定义作为主查询的子查询。

3.7 SQL like运算符区分大小写，但字符串的lower（）函数可用于执行不区分大小写的匹配。为了说明如何，编写一个查询，查找名称中包含字符串“sci”的部门作为子字符串，无论如何。

回答：

选择部门名称

从部门

其中lower（dept name）喜欢'％sci％'

3.8考虑SQL查询

练习13

分支机构（分支机构，分支机构，资产）

客户（客户名称，客户街道，客户城市）贷款（贷款编号，分支机构名称，金额）

借款人（客户名称，贷款号码）

帐户（帐号，分行名称，余额）

存款人（客户姓名，账号）

图3.1练习3.8和3.15的银行数据库。

选择p.a1

来自p，r1，r2

其中p.a1 = r1.a1或p.a1 = r2.a1

在什么条件下，前面的查询选择p.a 1的值是r 1还是r 2？仔细检查r 1或r 2中的一个可能为空的情况。

答案：当且仅当r1和r2都非空时，查询选择p.a1的值等于某个值r1.a1或r2.a1。如果r1和r2中的一个或两个为空，则p，r1和r2的笛卡尔乘积为空，因此查询结果为空。当然，如果p本身为空，则结果如预期的那样，即为空。

3.9考虑图3.19的银行数据库，其中主键是无线的。为此关系数据库构造以下SQL查询。

一个。查找拥有帐户但没有贷款的银行的所有客户。

湾找到与“史密斯”居住在同一街道和同一城市的所有客户的名称。

C。查找在银行拥有帐户并居住在“Harrison”的客户的所有分支机构的名称。

回答：

一个。查找拥有帐户但没有贷款的银行的所有客户。

（从存款人中选择客户名称除外）

（选择客户名称

从借款人）

上述选择可以选择具有不同的指定，而不会更改查询的结果。

湾找到与“史密斯”居住在同一街道和同一城市的所有客户的名称。

编写查询的一种方法如下。

14第3章SQL简介

选择F.customer名称

来自客户F加入客户S使用（客户街道，客户城市）

其中S.customer name ='Smith'

或者可以在where中指定连接条件

子句，而不是使用bf join .. using。

C。查找在银行拥有帐户并居住在“Harrison”的客户的所有分支机构的名称。

选择不同的分支名称

从帐户自然加入存款人自然加入客户

客户城市='哈里森'

像往常一样，可以通过在where子句中指定连接条件来替换自然连接操作。

3.10考虑图??的员工数据库，其中主键带下划线。为SQL中的每个查询提供一个表达式。

一个。查找为First Bank Corporation工作的所有员工的姓名和居住城市。

湾查找为First Bank Corporation工作且收入超过的所有员工的姓名，街道地址和居住城市

$ 10,000。

C。查找数据库中不适用于First Bank Corporation的所有员工。

d。查找数据库中收入超过Small Bank Corporation每位员工的所有员工。

即假设这些公司可能位于几个城市。查找位于Small Bank Corporation所在的每个城市的所有公司。

F。找到拥有最多员工的公司。

G。找到那些员工平均薪水高于First Bank Corporation平均薪水的公司。

回答：

员工（员工姓名，街道，城市）

工作（雇员姓名，公司名，工资）公司（公司名，市）

管理（员工姓名，经理姓名）

图3.20。员工数据库。

练习15

一个。查找为First Bank Corporation工作的所有员工的姓名和居住城市。

选择e.employee name，city

来自员工e，工作

其中w.company name ='First Bank Corporation'和

w.employee name = e.employee name

湾查找为First Bank Corporation工作且收入超过的所有员工的姓名，街道地址和居住城市

$ 10,000。

如果人们可以为几家公司工作，以下解决方案将仅列出那些每年从“第一银行公司”赚取超过10,000美元的人。

选择 \*

来自员工

员工姓名在哪里（从工作中选择员工姓名）

公司名称='第一银行公司'，工资> 10000）

与上一个查询的解决方案一样，我们也可以使用连接来解决这个问题。

C。查找数据库中不适用于First Bank Corporation的所有员工。

以下解决方案假设所有人都只为一家公司工作。

选择员工姓名

来自作品

公司名称/ ='First Bank Corporation'

如果允许人们出现在数据库中（例如在员工中）但没有出现在工作中，或者如果人们可能有多个公司的工作，则解决方案稍微复杂一些。

选择员工姓名

来自员工

员工姓名不在的地方（从工作中选择员工姓名）

公司名称='第一银行公司'）

d。查找数据库中收入超过Small Bank Corporation每位员工的所有员工。

16第3章SQL简介

以下解决方案假定所有人最多只能为一家公司工作。

选择员工姓名

来自作品

薪水>全部（从工作中选择薪水）

公司名称='小银行公司'）

如果人们可能为几家公司工作，我们希望考虑每个人的总收入，那么问题就更复杂了。它可以通过使用嵌套子查询来解决，但我们在下面说明如何使用with子句来解决它。

以总薪水为

（选择员工姓名，金额（工资）作为总薪水

来自作品

按员工姓名分组

）

从总薪水中选择员工姓名，总薪水>全部

（选择总薪水

从总薪水，工作

其中works.company name ='小银行公司'和

emp total salary.employee name = works.employee name

）

即假设这些公司可能位于几个城市。查找位于Small Bank Corporation所在的每个城市的所有公司。

最简单的解决方案使用包含在原始System R Sequel语言中的比较，但在后续SQL版本中不存在。

从公司T中选择T.company名称（选择R.city

来自R公司

其中R.company name = T.company name）

包含

（选择S.city

来自S公司

其中S.company name ='Small Bank Corporation'）下面是使用标准SQL的解决方案。

练习17

选择S.company名称

来自S公司

哪里不存在（（选择城市

来自公司

公司名称='小银行公司'）

除了

（选择城市

来自T公司

其中S.company name = T.company name））

F。找到拥有最多员工的公司。

选择公司名称

来自作品

按公司名称分组

有计数（不同的员工姓名）> =全部（从工作中选择计数（不同的员工姓名）

按公司名称分组）

G。找到那些员工平均薪水高于First Bank Corporation平均薪水的公司。

选择公司名称

来自作品

按公司名称分组

有平均（工资）>（选择平均（工资）

来自作品

公司名称='第一银行公司'）

3.11考虑图??的关系数据库。为SQL中的每个查询提供一个表达式。

一个。修改数据库，以便Jones现在居住在Newtown。

湾除非工资超过10万美元，否则给第一银行公司的所有经理加薪10％;在这种情况下，只提高3％。

回答：

一个。修改数据库，以便Jones现在居住在Newtown。

解决方案假设每个人只有一个元组

员工关系。

更新员工

设置city ='牛顿'

人名='琼斯'的地方

18第3章SQL简介

湾 除非薪水超过10万美元，否则给第一银行公司的所有经理加薪10％; 在这种情况下，只提高3％。

更新工作T.

设置T.salary = T.salary \* 1.03

其中T.employee名称（选择经理名称

来自管理）

和T.salary \* 1.1> 100000

和T.company name ='第一银行公司'

更新工作T.

设置T.salary = T.salary \* 1.1

其中T.employee名称（选择经理名称

来自管理）

和T.salary \* 1.1 <= 100000

和T.company name ='第一银行公司'

如果以相反的顺序执行，上述更新将给出不同的结果。 我们使用案例陈述给出了一个更安全的解决方案。

更新工作T.

设置T.salary = T.salary \*

（案件

当（T.salary \* 1.1> 100000）然后1.03

否则1.1

）

其中T.employee名称（选择经理名称

来自管理）和

T.company name ='First Bank Corporation'