《数据库系统》实验报告

第 2 次实验：使用 DB2 实现基本 SQL 操作

姓名：王琨

学号：091220103

09 级计算机科学与技术系三班

邮箱：woshibendan@stu.nju.edu.cn

时间：2011-11-6

目录

一、 实验目的

学习使用 DB2 设置数据库的完整性保护。

二、 实验要求

利用 DB2，创建一个新的数据库，在该数据库下创建如下表格：

1. 职工（姓名，工号，出生日期，家庭地址，年薪，管理员工号，所在部门编号）；
2. 部门（部门名称，部门编号，部门负责人的工号）；
3. 项目（项目名称，项目编号，所在城市，主管部门编号）；
4. 工作（职工工号，项目编号，工作时间）；
5. 家属（职工工号，家属的姓名，家属的性别）。

实现以下要求：

1. 表名和属性名用英文字符表示；
2. 表中各属性类型自行设计；
3. 分析该关系模式中存在的各种数据约束，并在创建表时体现出来；
4. 加入样本数据（自行设计），执行违反数据约束的操作，并观察结果；
5. 创建一个触发器，当职工参加一个新的项目时，年薪增加 2%；
6. 创建一个用户，将工作表的查询权限赋予该用户。

三、 实验分析

1. 表格说明

本次实验需要在数据库中创建五个表，以下给出五个表的属性名和相应的属性类型。

[1].职工表

表名	Employee（职工）						
属性名	姓名	工号	出生日期	家庭地址	年薪	管理员工号	所在部门编号
属性名	ename	eno	birthday	address	salary	ano	dno

属性类型	char(20)	int	date	char(20)	float	int	int

[2].部门表

表名	Department（部门）		
属性名	部门名称	部门编号	部门负责人工号
属性名	dname	dno	drno
属性类型	char(20)	int	int

[3].项目表

表名	Project（项目）			
属性名	项目名称	项目编号	所在城市	主管部门编号
属性名	pname	pno	pcity	Dno
属性类型	char(20)	int	char(20)	Int

[4].工作表

表名	Job（工作）		
属性名	职工工号	项目编号	工作时间
属性名	eno	pno	Hour
属性类型	int	int	Int

[5].家属表

表名	Family（家属）		
属性名	职工工号	家属姓名	家属的性别
属性名	eno	fname	Fsex
属性类型	int	char(20)	char(2)

2. 数据约束

PS.

每个人对表具有的语义关系理解可能不太一致，所以下面所给出的数据约束都是自定义的。

[1].Employee 表

主键约束：工号；

外键约束：所在部门编号；

非空性约束：姓名、工号、年薪、管理员工号、所在部门编号；

唯一性约束：工号；

表检查约束：无。

[2]. Department 表

主键约束：部门编号；

外键约束：无；（不是很确定部门负责人是否属于员工）

非空性约束：部门名称、部门编号、部门负责人编号；

唯一性约束：部门名称、部门编号；

表检查约束：无；

[3].Project 表

主键约束：项目编表；

外键约束：主管部门编号；

非空性约束：项目名称、项目编号、主管部门编号；

唯一性约束：无；

表检查约束：无；

[4].Job 表

主键约束：（职工编号、项目编号）；

外键约束：职工编号、项目编号；

非空性约束：职工编号；

唯一性约束：（职工编号、项目编号）；

表检查约束：工作时间 <= 8hours；

[5]. Family 表

主键约束：职工编号；

外键约束：职工编号；

非空性约束：职工编号、家属性名；

唯一性约束：职工工号；

表检查约束：家属的性别为男或者女或者空；

3. 数据样本

我们以“洛杉矶湖人队”队员为例，采集样本数据：

Employee（职工）

姓名	工号	出生日期	家庭地址	年薪	管理员工号	所在部门编号
Kobe	24	1978-8-23	America	15000000	100	1
Gasol	16	1980-7-6	Spain	8000000	100	1
Jackson	124	1945-9-17	America	8000000	100	2

Department（部门）

部门名称	部门编号	部门负责人工号
Basketball	1	100
Coach	2	100

Project（项目）

项目名称	项目编号	所在城市	主管部门编号
Game1	1	Los Angeles	1
Game2	2	Houston	1
Game3	3	Miami	1

Job（工作）

职工工号	项目编号	工作时间
24	1	2
16	2	2

Family（家属）		
职工工号	家属姓名	家属的性别
16	Marc	Null

四、 实验步骤

1. 创建表项

Done…

2. 添加数据样本

Done…

3. 测试数据约束

i. 非空性约束

在创建表“Project”过程中，我们设置该表的“项目名称”属性为“NOT NULL”，而“城市”属性为默认属性可以为“NULL”。

测试“NOT NULL”非空性约束：

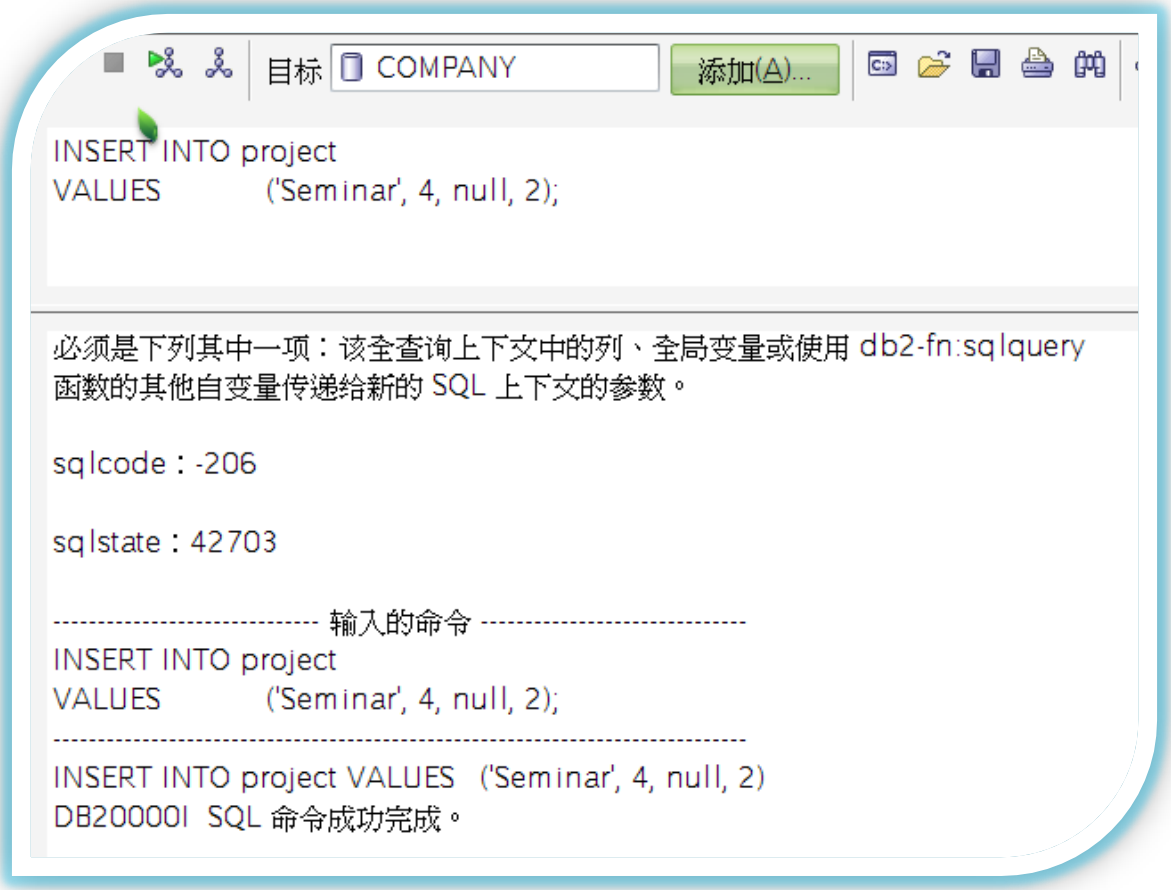
```
INSERT INTO project
VALUES    (null, 4, 'Washington', 2);
```

语句执行后得到如下的错误提示：

错误提示表示“COLNO=0”的那一列（pname）不允许为空值，即满足非空性约束。

测试“NULL”可空性约束：

```
INSERT INTO project
VALUES ('Seminar', 4, null, 2);
```



ii. 主键约束

主键约束隐含唯一性约束，它禁止在表的多列中出现重复值，从而逐渐可以成为每条记录的标识符。我们的测试想法是，在 Employee 表中插入一条记录，其键值已存在于表中：

```
INSERT INTO employee
VAULES ('Odom', 24, DATE('1978-8-23'), 'Los Angele', 15000000, 100, 1),
```

注意黄色高亮部分，由于记录“kobe，24，…”已经存在，所以此处违反主键约束。

根据 DB2 信息中心的错误提示，我们很容易知道出现了主键约束错误。

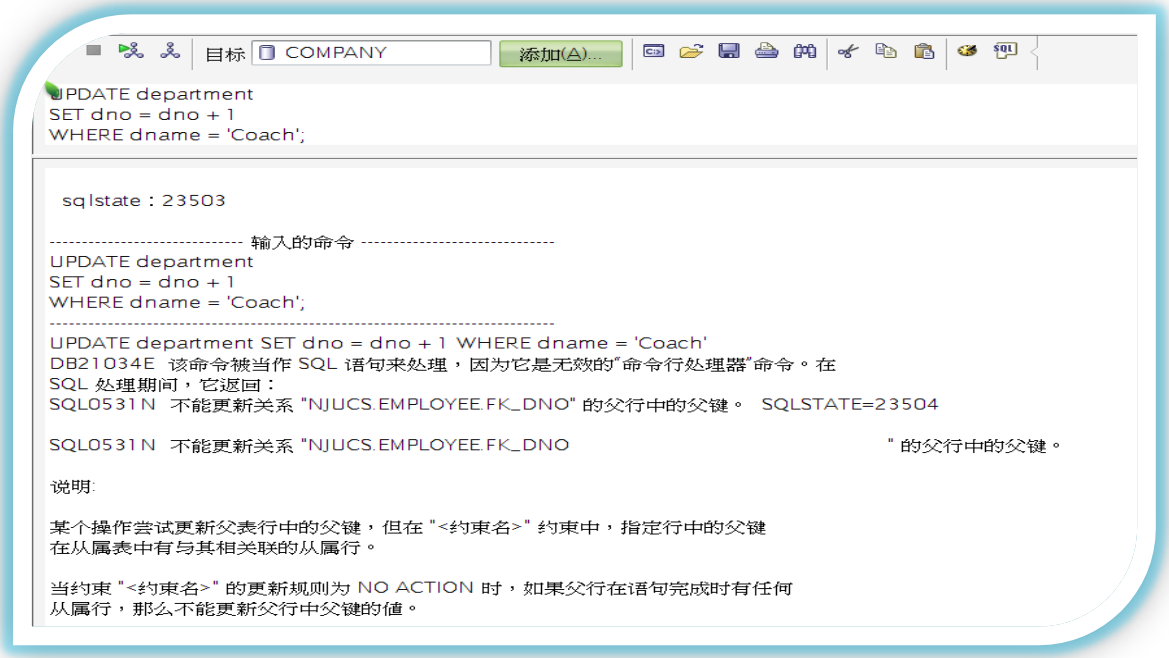
iii. 外键约束

在创建 Employee 表时，我们设置其属性“dno”为外键，直接引用“Department”中的 dno，Delete 参照操作是 CASCADE，UPDATE 参照动作是 NO ACTION。

根据外键约束规则，删除“Department”中的“Coach”部门将会导致“Employee”中的“Jackson”条目消失（没有教练这个职位，哪来的杰克逊。。。）；更新“Department”中的“Coach”无法执行（Update 参照操作为 NO ACTION）。

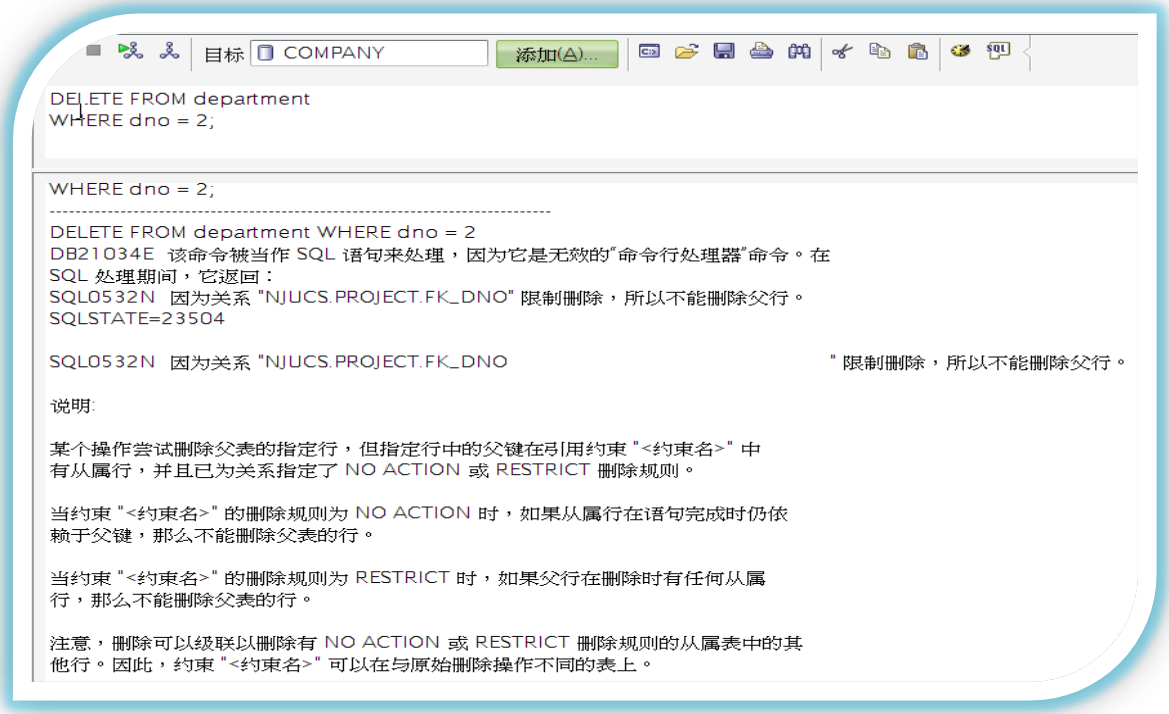
```
UPDATE department
SET dno = dno + 1
WHERE dname = 'Coach';
```

根据上面的分析，这条指令的执行将会出错：



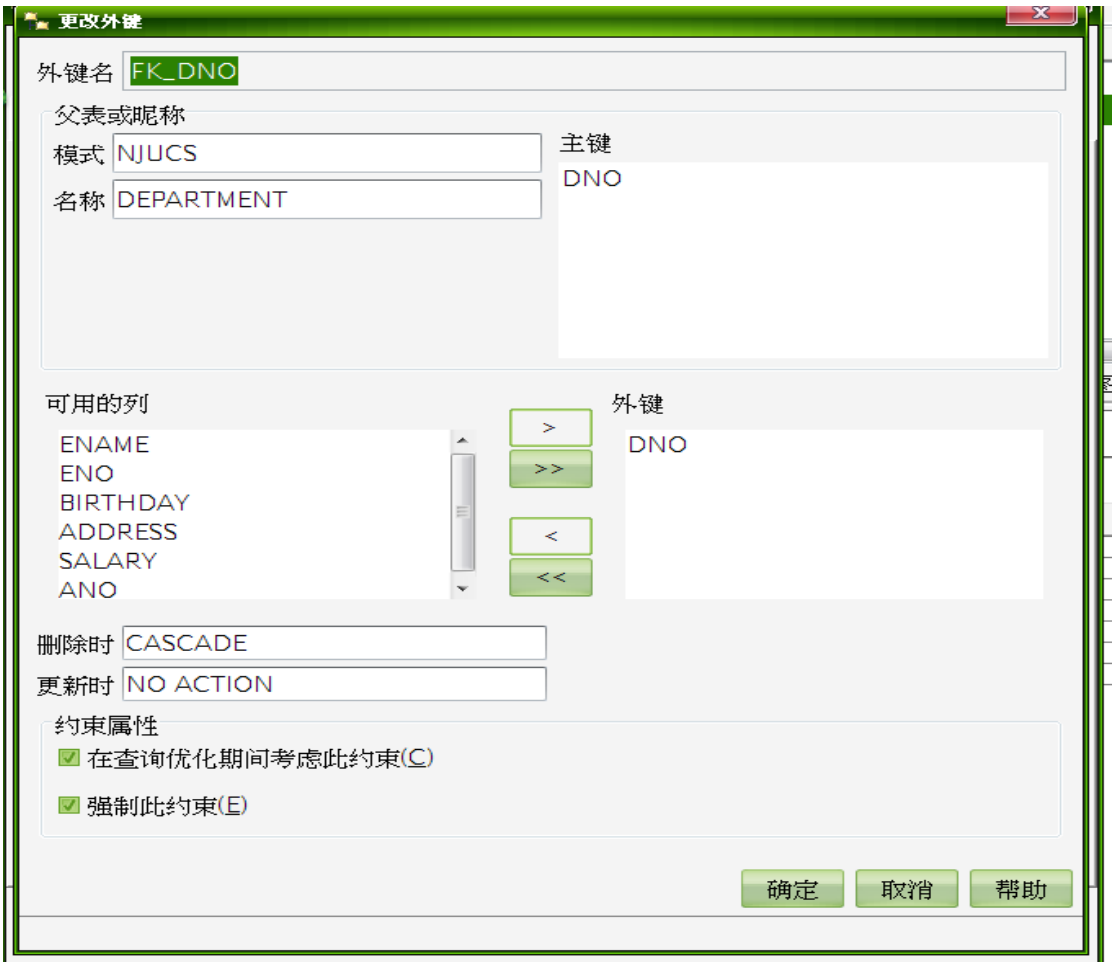
父键的更新操作无法进行。

```
DELETE FROM department
WHERE dno = 2;
```



BUG 出现了。由于 DELETE 的参照动作为 CASCADE，理论上删除操作是可以进行的， Employee 中相对应的元组也需要删除。然而 DB2

信息中心是直接报错。查看设置的 DELETE 参照动作：



确实设置的是 CASCADE。。。

iv. 表检查约束

需要指出的是，我们有一个 BUG 尚未解决：

Family 表中的 fsex 进行以下的表检查约束：

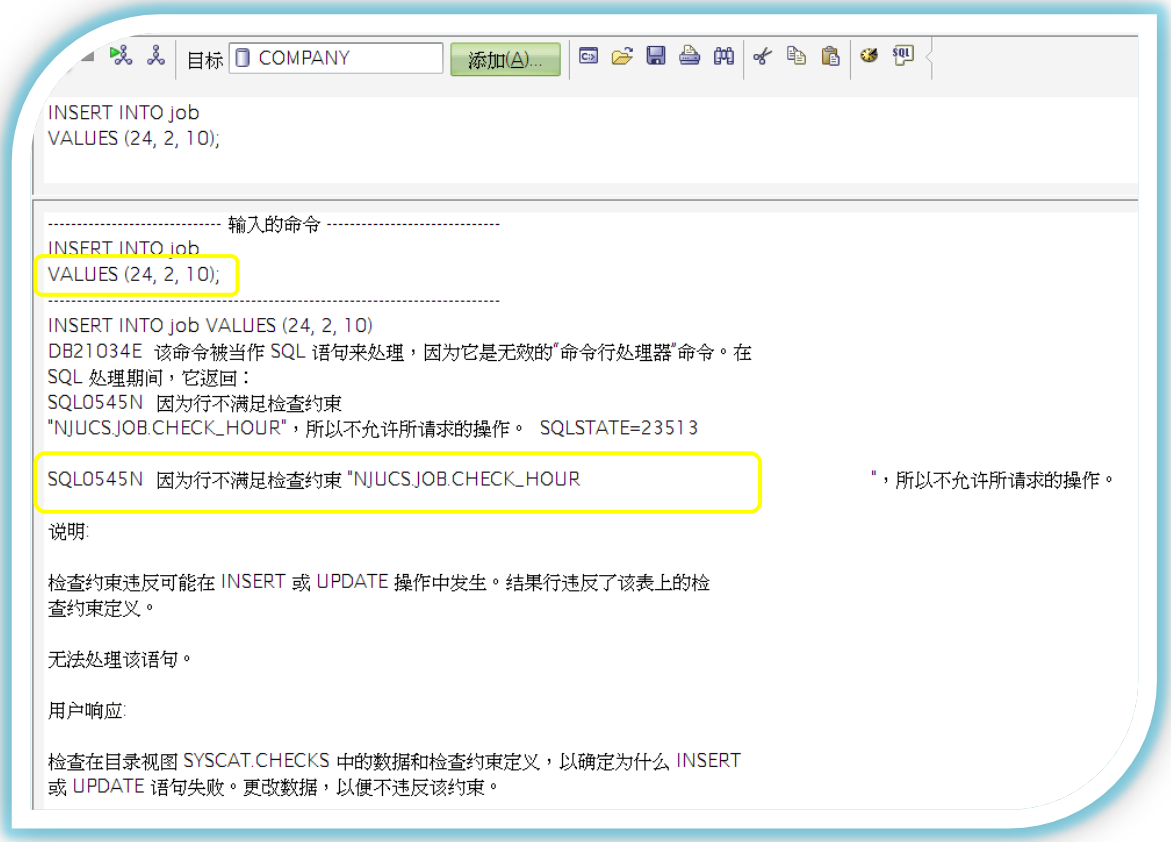
```
CONSTRAINT    check_fsex    CHECK  ( fsex = '男'  OR  fsex = '女'  OR  fsex = null )
```

即性别可以为空，如果不为空，那么只可以取值“男”、“女”。实际插入数据时，fsex = ‘男’无法成功。无论 fsex 设置为 1 字节还是 2 字节，都提示 fsex 长度过长。。。

没有办法，我们只好对 Job 中的“工作时间”进行约束：

```
CONSTRAINT     check_hour   CHECK  ( hour <= 8 )
```

工作最长时间不得超过 8 小时。写语句进行测试一下吧：



让菲尔杰克逊这个老头开研讨会 10 个小时还不如杀了他，所以数据库也不允许我们这样进行操作。

4. 实现触发器

```
create TRIGGER update_salary
    after INSERT on job              /*** 在job表插入一条新纪录之后更新employee表 */
    referencing NEW as new_item      /*** 将新建的插入项命名为 new_item */
    for each row                     /*** 注意row 和 statement 的区别 */
        update employee
        set salary = salary + salary*0.02
        where employee.eno = new_item.eno;
```

虽然菲尔杰克逊 10 个小时的研讨班扛不住，但是为了进行数据库触发器测试，勉强让他开会 3 个小时吧。。。所以我们会在 Job 中插入一条记录，然后触发器就开始工作了

```
INSERT INTO job
VALUES (124, 4, 3) // 表示菲尔参与项目 4 的工作，3 小时
```

数据插入前：

命令 查询结果 访问方案

采用定位 UPDATE 和定位 DELETE 编辑这些结果。使用"工具设置"笔记本来更改编辑格式。

ENAME	ENO	BIRTHDAY	ADDRESS	SALARY	ANO	DNO
Gasol	16	1980-7-6	Spain	8,160,000	100	1
Kobe	24	1978-8-23	Los Angele	15,300,000	100	1
Jackson	124	1945-9-17	America	10,000,000	100	2

数据插入后：

命令 查询结果 访问方案

采用定位 UPDATE 和定位 DELETE 编辑这些结果。使用"工具设置"笔记本来更改编辑格式。

ENAME	ENO	BIRTHDAY	ADDRESS	SALARY	ANO	DNO
Gasol	16	1980-7-6	Spain	8,160,000	100	1
Kobe	24	1978-8-23	Los Angele	15,300,000	100	1
Jackson	124	1945-9-17	America	10,200,000	100	2

10,200,000 = 10,000,000 * (1 + 2%)，OK！

5. 新建用户

测试“新建用户”的基本思路如下：

数据库 Company 的拥有者‘njucs’

————授权————》

新用户‘k41’（WITH GRANT OPTION）

————授权————》

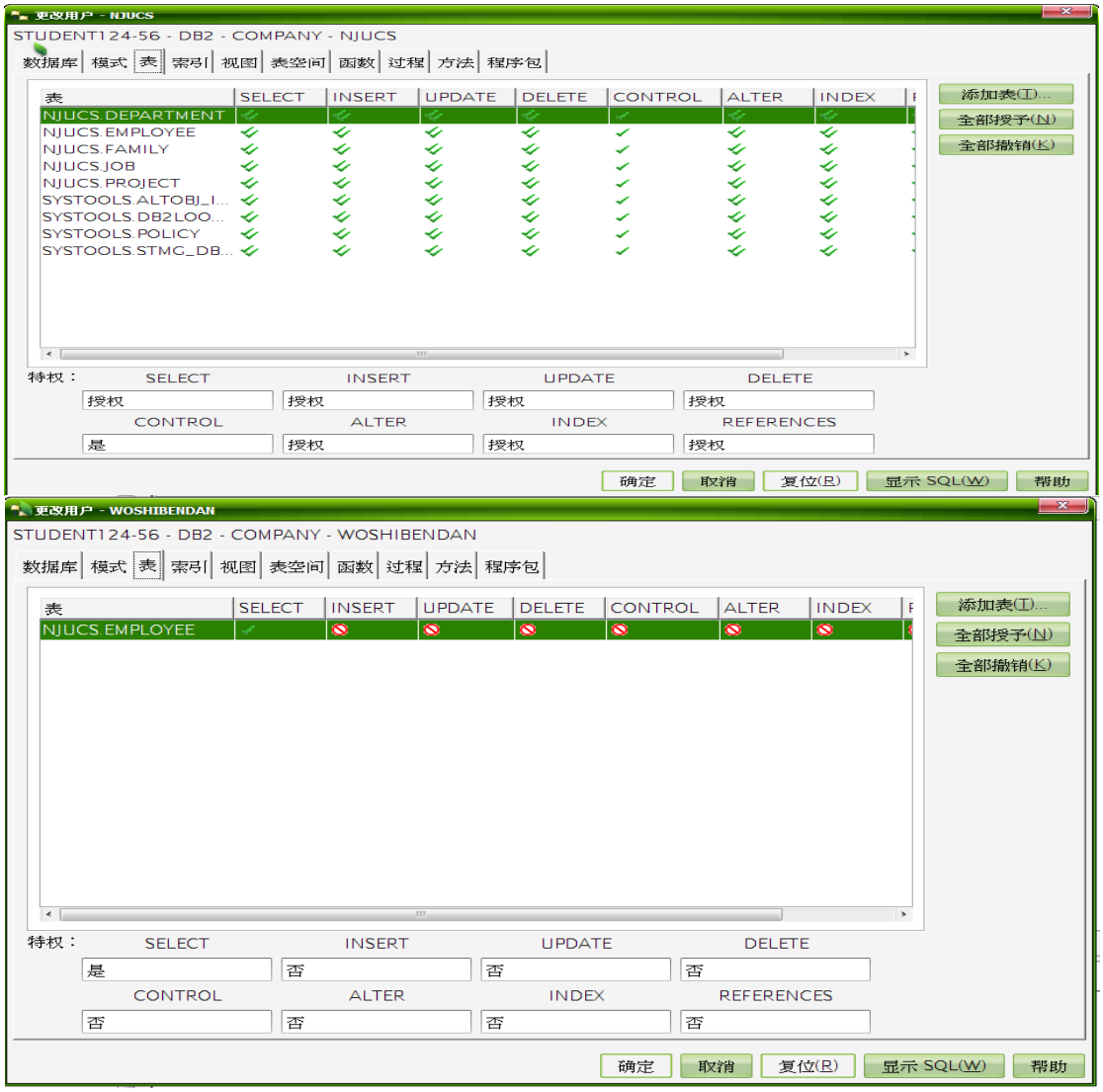
新用户‘woshibendan’ （WITHOUT GRANT OPTION）

PS.

不要对上面各种各样的怪名字好奇，机房电脑都这样。。。当让

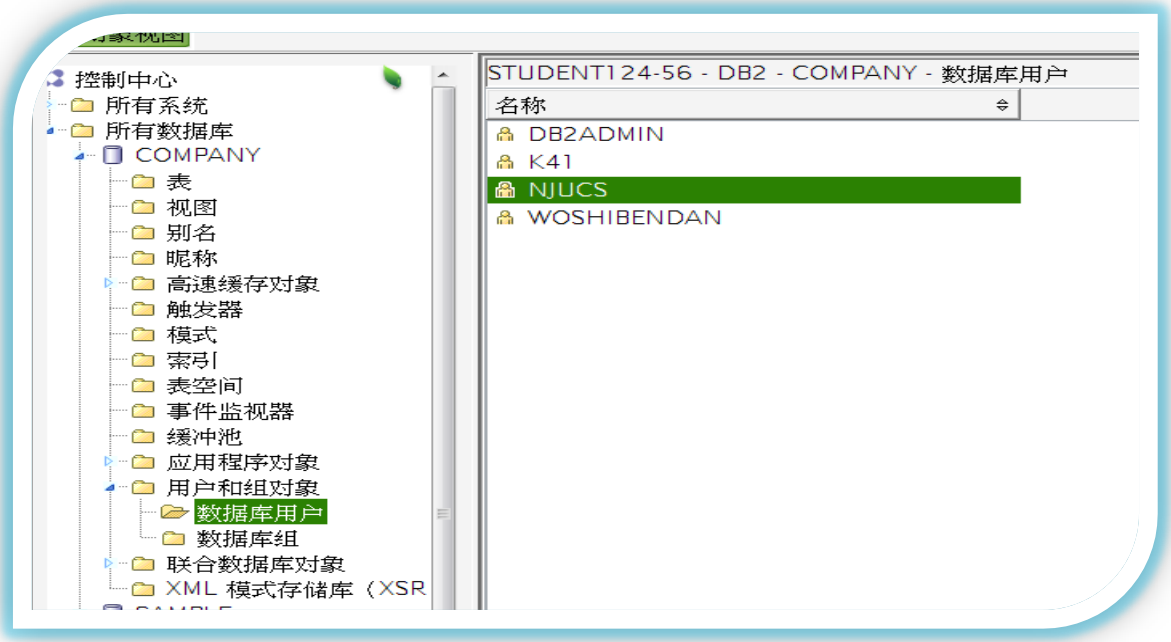
‘woshibendan’这个用户名实为测试临时创建的。。。

以下给出授权后‘njucs’和‘woshibendan’两个用户拥有的权限对比：



我们可以看到，‘woshibendan’仅仅拥有‘njucs’的 Company 数据库 Employee 的查询权利，而事实也是如此。以下我们分别用‘woshibendan’进行 Select 和 Insert 操作（需要说明的是，‘woshibendan’权利完全来自‘k41’），看看会出现怎样的结果。由于‘woshibendan’不具有 Insert 的权限，所以 Insert 操作报错，不可以将操作能力传递，但是它可以进行 Select 操作。

为证明所言非虚，下面给出拥有 Company 数据库的全部用户：



五、 问题 & 思考

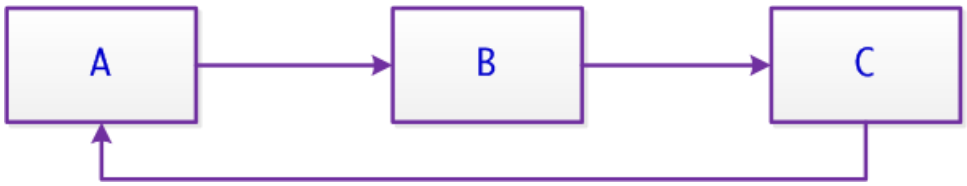
1. 触发器

注意 for each row和 for each statement 的区别。for each row 会逐行扫描给定表的所有项，所以 begin … end 语句块可能执行很多次；然而 for each statement 表示所有语句仅仅执行一次。在本次实验中，触发器 update_salary 使用 for each row 或者 for each statement 都是可行的，因为 eno 主键约束保证了搜索的唯一性。

2. 外键的 Insert 操作和外键环

```
***
表名                                    Employee（职工）
属性名      姓名      工号      出生日期    家庭地址      年薪      管理员工号    所在部门编号
属性名      ename     eno       birthday    address       salary    ano           dno
属性类型    char(8)   int       date        char(20)      float     int           int
*/
/*** 测试当外键引用不存在的时候，如何处理 */
INSERT INTO employee
VALUES ('Kobe', 24, DATE('1978-8-23'), 'Los Angele', 15000000, 100, 1);

--------------------------------------------------------------------------------
/*** 表名                                  Employee（职工） 属性名    姓名        工号        出生日期
DB21034E  该命令被当作 SQL 语句来处理，因为它是无效的"命令行处理器"命令。在
SQL 处理期间，它返回：
SQL0530N  FOREIGN KEY "NJUCS.EMPLOYEE.FK_DNO"
的插入或更新值不等于父表的任何父键值。  SQLSTATE=23503

SQL0530N  FOREIGN KEY "NJUCS.EMPLOYEE.FK_DNO                    " 的插入或更新值不等于父表的任何父键值。

说明:

正在设置对象表的外键中的值，但此值不等于父表的任何父键值。
```

Employee 表中的‘dno’外键引用 Department 中的‘dno’，如果 Employee 新插入的数据项‘dno’不为空并且在 Department 中找不到相应项，DB2 报错。即对于子表而言，外键的成功插入依赖于父表。是否存在这样的一类关系：



对于 A、B、C 三张表而言，他们相互外键依赖而构成环，根据实验中的例子，这样会导致表创建工作无法进行，因为我们无法在表 B 尚未定义的情况下创建表 A。这就是“外键环”（自定义），当然我们可以比较小心的设计 A、B、C 的表关系，使得不再出现环结构，然而这样可能会出现其他影响。

3. 外键约束的 Update 操作不支持 CASCADE

在实验过程中，我们发现外键约束中的 Update 参考动作不能为“CASCADE”，举一个例子说明这样的限制是正确的：

TableA

PID	NAME
1	PAR1
2	PAR2
3	PAR3

TableB

CID	CDESC	PID
51	51DESC	1
52	52DESC	2

我们对 TableA 进行 Updat 操作，pid = pid + 1（pid 与 name 均不是关键字），如果实现“CASCADE”，这会导致 CID = 51 的将映射到“PAR2”，CID=52 的映射到 “PAR3”，从而出现语义错误。

4. 外键约束的 Delete 操作 CASCADE 无效

这是本次实验的一个 BUG，我们在测试 Employee 中的外键约束时，发现虽然 dno 设置外键约束，Delete 参照动作为 CASCADE，但是实际运行时却不是如此。此问题需要等到后面解决。