Database-System Experiment-11

21307289 刘森元

1. 实验目的

学习用户自定义约束，并实践用户完整性，利用短语NOT NULL，UNIQUE，CHECK保证用户定义完整性。

2. 实验环境

Macbook Pro 2021 (Apple M1 Pro)

macOS Sonoma 14.1.1

PostgreSQL 15.4 (Homebrew)

zsh 5.9

3. 实验步骤

课内实验

1) 创建 Worker 表，并自定义两个约束U1 以及U2，其中 U1 规定 Name 字段唯一， U2 规定 sage(级别)字段的上限是28。(参考代码如下：)

```
Create Table Worker(
  Number char(5),
  Name char(8) constraint U1 unique,
  Sex char(1),
  Sage int constraint U2 check(Sage<= 28),
  Department char(20),
  constraint PK_Worker Primary Key(Number)
)
```

使用如下代码创建表 Worker

```
CREATE TABLE Worker (
    Number char(5),
    Name char(8) CONSTRAINT U1 UNIQUE,
    Sex char(1),
    Sage int CONSTRAINT U2 CHECK (Sage <= 28),
    Department char(20),
    CONSTRAINT PK_Worker PRIMARY KEY (Number)
);
```

有如下反馈

```
school=# CREATE TABLE Worker (
    Number char(5),
    Name char(8) CONSTRAINT U1 UNIQUE,
    Sex char(1),
    Sage int CONSTRAINT U2 CHECK (Sage <= 28),
    Department char(20),
    CONSTRAINT PK_Worker PRIMARY KEY (Number)
);
CREATE TABLE
school=# \d
                关联列表
 架构模式 |  名称  |  类型  |   拥有者
----------+--------+--------+-------------
 public   | worker | 数据表 | qiu_nangong
(1 行记录)

school=# \d worker
                数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u2" CHECK (sage <= 28)
```

可见表 Worker 成功创建

2）在 Worker 表中插入一条合法记录。(参考代码如下：)

```
INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00001', '李勇',
'M', 14, '科技部');
SELECT * FROM Worker;
```

使用如下代码插入合法记录

```
INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00001', '李勇',
'M', 14, '科技部');
SELECT * FROM Worker;
```

有如下反馈

```
school=# INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00001',
'李勇', 'M', 14, '科技部');
SELECT * FROM Worker;
INSERT 0 1
 number |    name    | sex | sage |       department
--------+------------+-----+------+-------------------------
 00001  | 李勇       | M   |   14 | 科技部
(1 行记录)
```

可见能成功插入数据

3）演示插入违反 U2 约束的例子，U2 规定元组的 sage 属性的值必须小于等于28。

使用如下代码插入数据

```
INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00002', '张三',
'F', 30, '人力资源部');
```

有如下反馈

```
school=# INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00002',
'张三', 'F', 30, '人力资源部');
ERROR:  new row for relation "worker" violates check constraint "u2"
描述:  Failing row contains (00002, 张三      , F, 30, 人力资源部               ).
```

这个插入操作会失败，因为违反了 U2 约束。

4）去除 U2 约束

使用如下代码去除约束

```
ALTER TABLE Worker
DROP CONSTRAINT U2;
```

有如下反馈

```
school=# ALTER TABLE Worker
DROP CONSTRAINT U2;
ALTER TABLE
school=# \d Worker
                数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
```

可见 U2 约束已去除

5）重新插入 3）中想要插入的数据，由于去除了 U2 约束，所以插入成功。

使用如下代码插入数据

```
INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00002', '张三',
'F', 30, '人力资源部');
```

有如下反馈

```
school=# INSERT INTO Worker(Number, Name, Sex, Sage, Department) VALUES ('00002',
'张三', 'F', 30, '人力资源部');
INSERT 0 1
school=# SELECT * FROM Worker;
 number |    name    | sex | sage |        department
--------+------------+-----+------+--------------------------
 00001  | 李勇       | M   |   14 | 科技部
 00002  | 张三       | F   |   30 | 人力资源部
(2 行记录)
```

可见已成功插入

6) 创建规则 rule_sex，规定插入或更新的值只能是 M 或 F，并绑定到 Worker 的 sex 字段。

通过如下代码创建规则

```
CREATE OR REPLACE RULE rule_sex AS
    ON INSERT TO Worker
    WHERE NEW.Sex NOT IN ('M', 'F')
    DO INSTEAD NOTHING;
```

有如下反馈

```
school=# CREATE OR REPLACE RULE rule_sex AS
    ON INSERT TO Worker
    WHERE NEW.Sex NOT IN ('M', 'F')
    DO INSTEAD NOTHING;
CREATE RULE
school=# \d Worker
                数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
规则:
    rule_sex AS
    ON INSERT TO worker
   WHERE new.sex <> ALL (ARRAY['M'::bpchar, 'F'::bpchar]) DO INSTEAD NOTHING
```

可见规则已成功添加

7）演示违反规则 rule_sex 的插入操作。

通过如下代码插入非法数据

```
INSERT INTO Worker (Number, Name, Sex, Sage, Department)
VALUES ('00003', '王五', 'X', 25, '销售部');
```

有如下反馈

```
school=# INSERT INTO Worker (Number, Name, Sex, Sage, Department)
VALUES ('00003', '王五', 'X', 25, '销售部');
INSERT 0 0
school=# SELECT * FROM Worker;
 number |    name    | sex | sage |        department
--------+------------+-----+------+--------------------------
 00001  | 李勇       | M   |   14 | 科技部
 00002  | 张三       | F   |   30 | 人力资源部
(2 行记录)
```

可见规则成功应用

自我实践

1）加入约束 U3，令 sage 的值大于等于0。

通过如下代码创建约束

```
ALTER TABLE Worker
ADD CONSTRAINT U3 CHECK (Sage >= 0);
```

有如下反馈

```
school=# ALTER TABLE Worker
ADD CONSTRAINT U3 CHECK (Sage >= 0);
ALTER TABLE
school=# \d Worker
                数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
```

```
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u3" CHECK (sage >= 0)
规则:
    rule_sex AS
    ON INSERT TO worker
   WHERE new.sex <> ALL (ARRAY['M'::bpchar, 'F'::bpchar]) DO INSTEAD NOTHING
```

可见约束已成功创建

2）加入规则 R2，确保插入的记录的 sage 值在1到100之间，并绑定到 sage 属性上。

使用如下代码创建规则

```
CREATE OR REPLACE RULE R2 AS
    ON INSERT TO Worker
    WHERE NEW.sage < 1 OR NEW.sage > 100
    DO INSTEAD NOTHING;
```

有如下反馈

```
school=# CREATE OR REPLACE RULE R2 AS
    ON INSERT TO Worker
    WHERE NEW.sage < 1 OR NEW.sage > 100
    DO INSTEAD NOTHING;
CREATE RULE
school=# \d Worker
                数据表 "public.worker"
    栏位    |     类型      | 校对规则 |  可空的  | 预设
------------+---------------+----------+----------+------
 number     | character(5)  |          | not null |
 name       | character(8)  |          |          |
 sex        | character(1)  |          |          |
 sage       | integer       |          |          |
 department | character(20) |          |          |
索引:
    "pk_worker" PRIMARY KEY, btree (number)
    "u1" UNIQUE CONSTRAINT, btree (name)
检查约束限制
    "u3" CHECK (sage >= 0)
规则:
```

```
 r2 AS
 ON INSERT TO worker
WHERE new.sage < 1 OR new.sage > 100 DO INSTEAD NOTHING
 rule_sex AS
 ON INSERT TO worker
WHERE new.sex <> ALL (ARRAY['M'::bpchar, 'F'::bpchar]) DO INSTEAD NOTHING
```

可见规则已成功创建

4. 实验心得

本次实验主要是学习用户自定义约束并实践用户完整性。在实验中，通过创建约束和规则的操作，验证了它们的有效性。通过实践操作，加深了对用户自定义约束和完整性的理解，并掌握了相关的操作方法。