Database-System Experiment-7

21307289 刘森元

1. 实验目的

熟悉SQL语言支持的有关视图的操作，能够熟练使用SQL语句来创建需要的视图，对视图进行查询和取消视图。

2. 实验环境

Macbook Pro 2021 (Apple M1 Pro)

macOS Ventura 13.5.2

PostgreSQL 15.4 (Homebrew)

zsh 5.9

3. 实验内容

1. 定义常见的视图形式，包括：
    - 行列子集视图。
    - WITH CHECK OPTION的视图。
    - 基于多个基表的视图。
    - 基于视图的视图。
    - 带表达式的视图。
    - 分组视图。
2. 通过实验考察WITH CHECK OPTION这一语句在视图定义后产生的影响，包括对修改操作、删除操作、插入操作的影响。
3. 讨论视图的数据更新情况，对子行列视图进行数据更新。
4. 使用DROP语句删除一个视图，由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须显式删除。同样的原因，删除基表时，由该基表导出的所有视图定义都必须显式删除。

4. 实验步骤

完整代码见附件 *Experiment-7.sql*

使用命令 `\i Experiment-7.sql` 以运行脚本。

清空数据库并重新导入

由于本次试验涉及到增删，若重复操作初始数据库内容会不一致，所以该项操作是必要的。

```
-- 清空数据库并重新导入
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;
\cd '/Users/qiu_nangong/Documents/Github/Database-System/Experiment-7'
\i STUDENTS.sql
\i TEACHERS.sql
\i COURSES.sql
\i CHOICES.sql
```

课内实验

创建一个行列子集视图（视图名为CS），给出选课成绩合格的学生的编号，所选课程号和该课程成绩

```
CREATE VIEW CS AS
SELECT sid, cid, score
FROM CHOICES
WHERE score ⩾ 60;
```

创建基于多个基表的视图(视图名为SCT)，这个视图由学生姓名和其所选修的课程名及讲授该课 程的教师姓名构成

```
CREATE VIEW SCT AS
SELECT STUDENTS.sname, COURSES.cname, TEACHERS.tname
FROM CHOICES
JOIN STUDENTS ON CHOICES.sid = STUDENTS.sid
JOIN COURSES ON CHOICES.cid = COURSES.cid
JOIN TEACHERS ON CHOICES.tid = TEACHERS.tid;
```

创建带表达式的视图，由学生姓名、所选课程名和所有课程成绩都比原来多5分这几个属性组成

```
CREATE VIEW SCG AS
SELECT STUDENTS.sname, COURSES.cname, CHOICES.score + 5 AS new_score
FROM CHOICES
JOIN STUDENTS ON CHOICES.sid = STUDENTS.sid
JOIN COURSES ON CHOICES.cid = COURSES.cid;
```

创建分组视图，将学生的学号及其平均成绩定义为一个视图

```
CREATE VIEW AverageScore AS
SELECT sid, AVG(score) AS average_score
FROM CHOICES
GROUP BY sid;
```

创建一个基于视图的视图，基于(1)中建立的视图，定义一个包括学生编号，学生所选课程数目和平均成绩的视图

```
CREATE VIEW StudentSummary AS
SELECT CS.sid, COUNT(CS.cid) AS course_count, AverageScore.average_score
FROM CS
JOIN AverageScore ON CS.sid = AverageScore.sid
GROUP BY CS.sid, AverageScore.average_score;
```

查询所有选修课程Software Engineering的学生姓名

```
SELECT sname
FROM SCT
WHERE cname = 'software engineering';
```

插入元组(600000000,823069829,10010,59)到视图CS中。若是在视图的定义中存在WITH CHECK OPTION子句对插入操作有什么影响?

```
INSERT INTO CS (sid, cid, score)
VALUES ('600000000', '10010', 59);
```

它会检查插入的数据是否满足视图的过滤条件。如果不满足条件，插入操作将被拒绝

```
psql:Experiment-7.sql:62: ERROR:  null value in column "no" of relation "choices"
violates not-null constraint
描述:  Failing row contains (null, 600000000, null, 10010, 59).
```

将视图CS（包含定义WITH CHECK OPTION)中，所有课程编号为10010的课程的成绩都减去5分。这个操作数据库是否会正确执行，为什么?如果加上5分(原来95分以上的不变)呢?

```
UPDATE CS
SET score = score + 5
WHERE cid = '10010';
```

这个操作不会被执行，因为它会导致视图中的数据不再满足过滤条件。

在视图CS（包含定义WITH CHECK OPTION)删除编号为804529880学生的记录，会产生什么结果?

这个操作不会被执行，因为删除后视图中的数据将不再满足过滤条件。

取消视图SCT和视图CS

```
DROP VIEW SCT CASCADE;
DROP VIEW CS CASCADE;
```

自我实践

定义选课信息和课程名称的视图VIEWC

```
CREATE VIEW VIEWC AS
SELECT CHOICES.sid, CHOICES.tid, CHOICES.cid, CHOICES.score, COURSES.cname
FROM CHOICES
JOIN COURSES ON CHOICES.cid = COURSES.cid;
```

定义学生姓名与选课信息的视图VIEWS

```
CREATE VIEW VIEWS AS
SELECT CHOICES.sid, CHOICES.tid, CHOICES.cid, CHOICES.score, STUDENTS.sname
FROM CHOICES
JOIN STUDENTS ON STUDENTS.sid = CHOICES.sid;
```

定义年级低于1998的学生的视图S1(SID,SNAME,GRADE)

```
CREATE VIEW S1 AS
SELECT sid, sname, grade
FROM STUDENTS
WHERE grade < 1998;
```

查询学生为“uxjof”的学生的选课信息

```
SELECT * FROM VIEWS
WHERE sname = 'uxjof';
```

查询选修课程“UML”的学生的编号和成绩

```
SELECT * FROM VIEWC
WHERE cname = 'uml';
```

向视图S1插入记录(“60000001,Lily,2001”)

```
INSERT INTO S1 (sid, sname, grade)
VALUES ('60000001', 'Lily', 2001);
```

定义包括更新和插入约束的视图S1，尝试向视图插入记录(“60000001,Lily,1997"),删除所有年级为1999的学生记录，讨论更新和插入约束带来的影响

```
DROP VIEW S1;

CREATE VIEW S1 AS
SELECT sid, sname, grade
FROM STUDENTS
WITH CHECK OPTION;

INSERT INTO S1 (sid, sname, grade)
VALUES ('60000001', 'Lily', 1997);

DELETE FROM S1
WHERE grade = 1999;
```

会有如下错误信息

```
psql:Experiment-7.sql:125: ERROR:  duplicate key value violates unique constraint
"students_pkey"
描述:  Key (sid)=(60000001 ) already exists.
psql:Experiment-7.sql:128: ERROR:  update or delete on table "students" violates
foreign key constraint "fk_choices_students" on table "choices"
描述:  Key (sid)=(800008565) is still referenced from table "choices".
```

这是由于该视图具有更新和插入约束，而原表中已存在该学生键。

在视图VIEWS中将姓名为“uxjof”的学生的选课成绩都加上5分

```
UPDATE VIEWS
SET score = score + 5
WHERE sname = 'uxjof';
```

有如下错误信息

```
psql:Experiment-7.sql:134: ERROR:  cannot update view "views"
描述:  Views that do not select from a single table or view are not automatically
updatable.
提示:  To enable updating the view, provide an INSTEAD OF UPDATE trigger or an
unconditional ON UPDATE DO INSTEAD rule.
```

这是由于该视图与原表具有更新和插入约束。

取消以上建立的所有视图

```
DROP VIEW VIEWC CASCADE;
DROP VIEW VIEWS CASCADE;
DROP VIEW S1 CASCADE;
```

5． 实验心得

在这个实验中，我学习了

- 如何创建和操作数据库中的视图。视图是基于一个或多个表的查询结果，它提供了一种虚拟的表格，可以简化复杂的查询操作，并提供更方便的数据访问方式。
- 如何在视图中使用更新和插入约束，以确保数据的完整性和一致性。通过启用约束，可以限制对视图的更新和插入操作，以满足特定的业务需求。
- 如何使用视图进行数据查询和操作。通过查询视图，可以从不同角度和维度分析和提取数据，而无需直接操作基础表。
- 深入了解了视图的概念和用法，并掌握了在数据库中创建、操作和使用视图的技巧。

视图是数据库中非常强大和灵活的工具，可以帮助我们更高效地管理和访问数据。