

《Introduction to Database System A*》 Experiment 5

contents: data control (DDL) 1. constraints(Priamry key ,not null , check , foreign key) 2.trigger 3. index	Lab:10-409 Time :2024.12.6
Instructor: liuli	
Stu_id 2023337621159	Stu_name 代翔
Software Environment : Windows/linux/macOS + postgresql/MYSQL/...	
<p>Purpose:</p> <ol style="list-style-type: none"> 1. To understand database constrains better by implementing constrains declaration (primary key, foreign key, not null, check etc.) on the running database. 2. To implement the trigger in the database; 3. To understand database indexes; <p>notation:</p> <p>1. Define constraints</p> <ul style="list-style-type: none"> ✓ Declare not null Constraint <ul style="list-style-type: none"> ■ declare not null constraint when we declare an attribute <pre>CREATE TABLE <table_name>(Attributes1 TYPE [DEFAULT value][NOT NULL][<constraints >], Attributes2 TYPE [DEFAULT value][NOT NULL][<constraints >], [constraints])</pre> ✓ Declare SQL CHECK Constraint on CREATE TABLE <pre>CREATE TABLE <name> (<list of elements> CHECK(<conditions>));</pre> ✓ Declare primary key Constraint <ul style="list-style-type: none"> ■ declare primary key constraint when we declare an attribute ■ Make an additional declaration that says a particular attribute or set of attributes form a key.eg. <pre>CREATE TABLE <name> (<list of elements> PRIMARY KEY(<attributes>) Or UNIQUE(<attributes>));</pre> 	

- ✓ Declare **Foreign key** Constraint
(before declare the foreign key, the referenced attribute or attributes set must have been defined as the primary key in the referenced table)

```
CREATE TABLE <name> (  
    <list of elements>  
    Foreign key(<attributes>) REFERENCES <table>(<attributes>)  
    [ON UPDATE SET NULL|CASCADE]  
    [ON DELETE SET NULL|CASCADE]  
);
```

Example:

```
CREATE TABLE movieexec (  
    name char(30),  
    address varchar(100) ,  
    cert int(11) CHECK(cert>=1) ,  
    netWorth int(11) ,  
    PRIMARY KEY (cert)  
)  
CREATE TABLE Studio (  
    name CHAR(30) PRIMARY KEY,  
    address VARCHAR(255) NOT NULL,  
    code INT CHECK(code>30000),  
    presC INT DEFAULT 0,  
    FOREIGN KEY(presC) REFERENCES MovieExec(cert)  
        ON DELETE CASCADE  
        ON UPDATE SET NULL  
);
```

2. Define or drop triggers

(1)define trigger function

```
CREATE FUNCTION triggerfunctionname()  
RETURNS TRIGGER AS  
$$  
BEGIN  
    Sqlstatement;  
    RETURN NEW|OLD|NULL;  
END;  
$$  
LANGUAGE plpgsql  
triggerfunctionname()
```

(2)define trigger

```
CREATE TRIGGER trigger_name  
BEFORE|AFTER INSERT|DELETE|UPDATE ON tbl_name  
[FOR EACH ROW] [FOR EACH STATEMENT]  
Execute function
```

(2)drop trigger function and trigger

DROP function triggerfunctionname() cascade

3. Define or drop index

(1) define index

CREATE INDEX <INDEXname> **ON** <table_name>(<list of elements>);

(2) drop index

DROP INDEX <INDEXname>

Content:

Handing your **PDF** answer file using the file name as exp5_ID(ID should be replaced by your own student ID);

Experiment contents:

Background description is shown in project1:

Schema for our running example:

Movies(title,year,length,movietype, studio**name**, producer**C**)

movieStar(name, address,gender,birthdate)

starsIn(movietitle, movieyear,starname)

Studio(name, address,pres**C**)

Movieexec(name, address,cert,networth)

list your answer in the last page. Handing your PDF answer file with the file name as exp5_ID.

part1. : Create new database as “my_Movies_datacontrol”;Write Create table statement with SQL;be sure to include the **constraints** in the following list:(To define the foreign key constraint,you should define primary key in referenced table first);

- 1.) declare **primary key** for each relation based on the schema above;
- 2.) the name in **movieexec** can not be null and the identification number(cert) of movie executive are in the range[1,20000];
- 3.) the gender in **Moviestar** can only either be ‘F’ or ‘M’;
- 4.) declare the foreign key constraints for the database such that each movie in **starsin** must appear in table **movies**; (choose policies such that the update of movie information from **movies** leads to same changes in the table **starsin** ; deletion of movies from **movies** leads to null values for the corresponding tuples in table starsin)

Write the SQL statement to create tables for the database(create table movieexec and moviestar first ,and then studio,movies,starsin)

load data from our experimental document.

part2. explain what happens, if anything, to maintain the referential integrity constraints in the following cases:

1. Update title information of a movie in movies relation from the old value to a new value(the old value in movies relation is also in starsin relation);
2. Delete a tuple in movies relation with the movie information also in starsin relation.

part3. define two triggers for movieexec relation so that they work as the functions in the following: (1)after a tuple in movieexec relation is updated by cert column,let the system make the same change on the same number of person in relation studio;(2)after delete a tuple in movieexec relation, let the same number of person in relation studio become null value.

Write the SQL statement

(observe:in movieexec, for the executive with cert value as 199,increase cert column' value by 100 ,query In studio,what's the change;delete the executive whose cert values as 299, query in studio,what's the change)

part4. Define index

declare indexes on the following attributes or combination of attributes.

(1)Movies(title,year);

(2)Movies(Studio name);

Part 1: Create table statements including constraints

```
CREATE DATABASE `my_Movies_datacontrol`;  
USE `my_Movies_datacontrol`;
```

```
CREATE TABLE `Movieexec`(  
  `name` VARCHAR(30) NOT NULL,  
  `address` VARCHAR(255),  
  `cert` INT PRIMARY KEY CHECK(`cert` BETWEEN 1 AND 20000),  
  `networth` INT  
);
```

```
CREATE TABLE `movieStar`(  
  `name` VARCHAR(30) PRIMARY KEY,  
  `address` VARCHAR(255),  
  `gender` VARCHAR(1) CHECK(`gender` IN ('F','M')),  
  `birthdate` DATE  
);
```

```
CREATE TABLE `Studio`(  
  `name` VARCHAR(30) PRIMARY KEY,  
  `address` VARCHAR(255),  
  `presC` INT  
);
```

```
CREATE TABLE `Movies`(  
  `title` VARCHAR(30),  
  `year` INT,  
  `length` INT,  
  `movietype` VARCHAR(30),  
  `studioName` VARCHAR(30),  
  `producerC` INT,  
  PRIMARY KEY(`title`,`year`),  
  FOREIGN KEY (`studioName`) REFERENCES `Studio`(`name`)  
);
```

```
CREATE TABLE `starsIn`(  
  `movietitle` VARCHAR(30),  
  `movieyear` INT,  
  `starname` VARCHAR(30),  
  FOREIGN KEY (`movietitle`,`movieyear`) REFERENCES `Movies`(`title`,`year`) ON UPDATE  
  CASCADE ON DELETE SET NULL,  
  FOREIGN KEY (`starname`) REFERENCES `movieStar`(`name`)
```

);

Part 2:

1. CREATE TRIGGER tri_update_movies AFTER
2. UPDATE ON `Movies`
3. FOR EACH ROW UPDATE `starsIn`
4. SET `movietite`=NEW.title
5. WHERE `movietite`=OLD.title AND `movieyear` = OLD.year;
6. CREATE TRIGGER tri_delete_movies
7. AFTER DELETE ON `Movies`
8. FOR EACH ROW DELETE FROM `starsIn`
9. WHERE `movietite`=OLD.title AND `movieyear` = OLD.year;

Part 3:declare trigger and trigger function

```
CREATE TRIGGER tri_update_movieexec
AFTER UPDATE ON `Movieexec`
FOR EACH ROW
BEGIN
    IF OLD.cert<>NEW.cert THEN
        UPDATE `Studio` SET `presC`=NEW.cert
        WHERE `presC`=OLD.cert;
    END IF;
END $$
```

DELIMITER ;

DELIMITER \$\$

```
CREATE TRIGGER tri_delete_movieexec
AFTER DELETE ON `movieexec`
FOR EACH ROW
BEGIN
    UPDATE `Studio`
    SET `presC`=NULL
    WHERE `presC`=OLD.cert;
END $$
```

DELIMITER ;

```
UPDATE `movieexec`
SET `cert`= 299
WHERE `cert`=199;
```

```
DELETE FROM `movieexec`
WHERE `cert`=299;
```

	name	address	presC
▶	Disney	Disney Boulevard	222
	Fox	Fox Boulevard	199
	huaxia	Boulevard	100
	MGM	MGM Boulevard	123
	Paramount	Paramount Boulevard	555
	USA Entertainm	USA Entertainm Boulevard	333
*	NULL	NULL	NULL

	name	address	cert	networth
▶	Huangjianxin	NULL	100	NULL
	Calvin Coolidge	Fast Lane	123	20000000
	Merv Griffin	Riot Rd.	199	112000000
	Stephen Spielberg	123 ET road	222	100000000
	Ted Turner	Turner Av.	333	125000000
	George Lucas	Oak Rd.	555	200000000
	Jane Fonda	Turner Av.	567	200000000
*	NULL	NULL	NULL	NULL

	name	address	cert	networth
▶	Huangjianxin	NULL	100	NULL
	Calvin Coolidge	Fast Lane	123	20000000
	Stephen Spielberg	123 ET road	222	100000000
	Merv Griffin	Riot Rd.	299	112000000
	Ted Turner	Ted Turner	333	125000000
	George Lucas	Oak Rd.	555	200000000
	Jane Fonda	Turner Av.	567	200000000
*	NULL	NULL	NULL	NULL

	name	address	presC
▶	Disney	Disney Boulevard	222
	Fox	Fox Boulevard	299
	huaxia	Boulevard	100
	MGM	MGM Boulevard	123
	Paramount	Paramount Boulevard	555
	USA Entertainm	USA Entertainm Boulevard	333
*	NULL	NULL	NULL

	name	address	cert	networth
▶	Huangjianxin	NULL	100	NULL
	Calvin Coolidge	Fast Lane	123	20000000
	Stephen Spielberg	123 ET road	222	100000000
	Ted Turner	Turner Av.	333	125000000
	George Lucas	Oak Rd.	555	200000000
	Jane Fonda	Turner Av.	567	200000000
*	NULL	NULL	NULL	NULL

movieexec 1 x Apply

	name	address	presC
▶	Disney	Disney Boulevard	222
	Fox	Fox Boulevard	NULL
	huaxia	Boulevard	100
	MGM	MGM Boulevard	123
	Paramount	Paramount Boulevard	555
	USA Entertainm	USA Entertainm Boulevard	333
*	NULL	NULL	NULL

Part 4:declare indexes

- 1.CREATE INDEX movie_ty ON `Movies`(`title`,`year`);
- 2.CREATE INDEX movie_studio ON `Movies`(`studio`name`);