实验八 完整性约束与触发器

姓名	学号	学院	日期
臧祝利	202011998088	人工智能学院	

实验目的

- 了解什么是触发器；
- 掌握创建触发器、调用触发器、删除触发器的方法；
- 掌握触发器的使用技巧。

实验内容

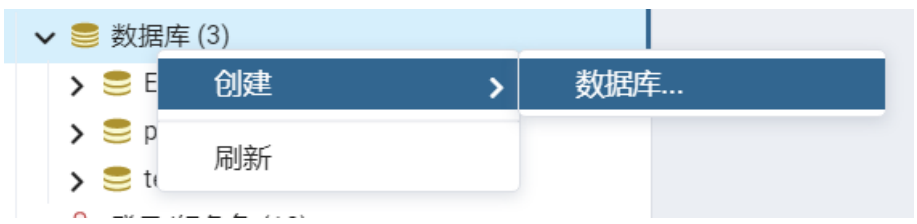
- 对于表的 `insert`、`delete`、`update` 操作，创建触发器函数并使用 `create trigger` 语句调用 `before`、`after` 触发器，对表中数据进行修改或抛出异常；
- 查看和删除触发器。

实验思路

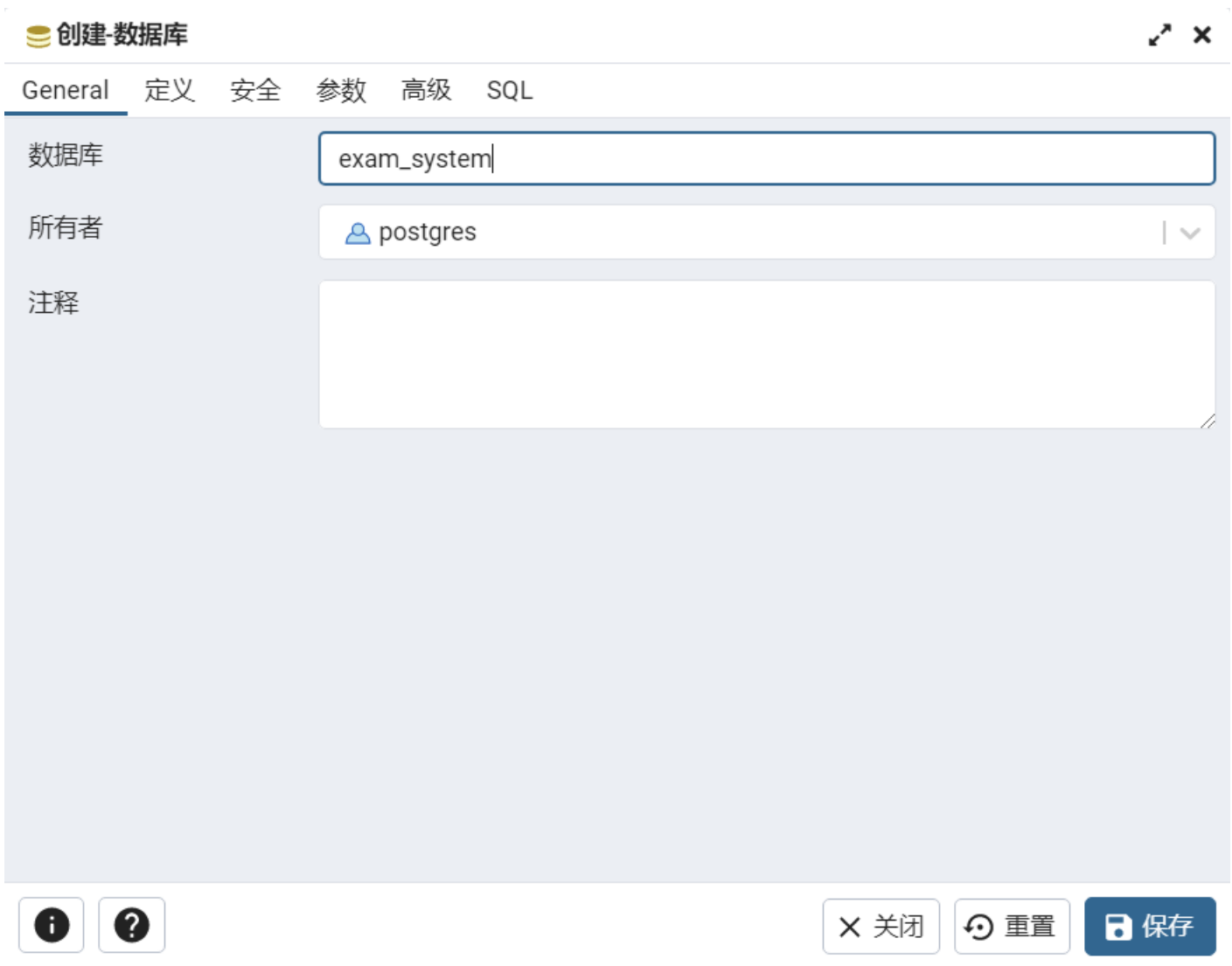
在 `exam_system` 数据库中进行以下触发器相关的操作:

设计 `exam_system` 数据库

Step1. 右击数据库，点击创建>数据库；



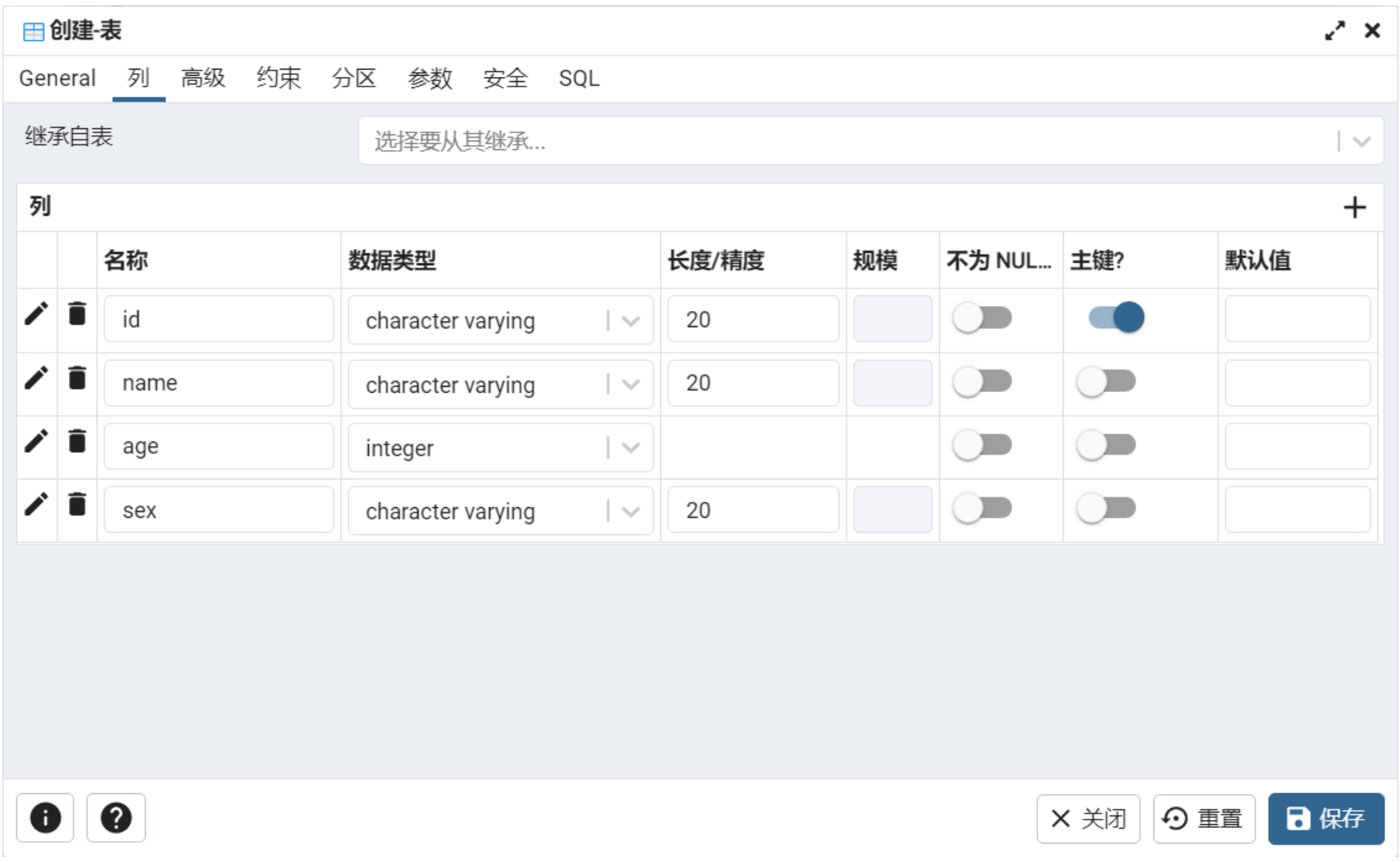
Step2. 输入数据库名称，点击保存；



Step3：建立 `student` 表，定义其中有 `id`，`name`，`age`，`sex` 属性；

在数据库中找到【架构】，在【表】处右击，新建表;

输入以下属性，设置 `id` 为 主键 ；保存即创建完毕;



Step4.：创建 `stuexam` 表，包括 `eeid`，`major`，`grade` 属性，其中设置 `eeid` 为主键;

使用代码创建：

```
create table stuexam(
    eeid character varying(20),
    major char(20) not null,
    grade smallint not null,
    CONSTRAINT stuexam_pkey PRIMARY KEY (eeid)
);
```

实验任务

- 建立 `INSERT` 触发器，若插入 `student` 表的记录学号长度不为 `10` 位，提示“学号格式错误！”（提示：可使用 `CHAR_LENGTH()` 获取字符串长度）
    - 输入以下代码，创建触发器函数 `idcheck()`

```
CREATE FUNCTION idcheck()
RETURNS TRIGGER AS $idcheck$
BEGIN
    IF (CHAR_LENGTH(new.id)<>10) THEN
        RAISE EXCEPTION '学号格式错误! ';
        RETURN NULL;
    END IF;
    RETURN NEW;
END;
$idcheck$ LANGUAGE plpgsql;
```

    - 创建触发器 `id_insert`

      代码如下：

```
CREATE TRIGGER id_insert
BEFORE INSERT ON student
FOR EACH ROW
EXECUTE PROCEDURE idcheck();
```

- 尝试进行数据插入，插入数据如下：

```
INSERT INTO student values('2011998088','Zang',20,'male');
INSERT INTO student values('2011998099','Luo',20,'male');
INSERT INTO student values('2011998073','Zhang',19,'female');
```

插入成功；

	id [PK] character varying (20)	name character varying (20)	age integer	sex character varying (20)
1	2011998073	Zhang	19	female
2	2011998088	Zang	20	male
3	2011998099	Luo	20	male

插入数据为：

```
INSERT INTO student values('202011998072','Zhao',19,'female');
INSERT INTO student values('2011998079','Li',20,'male');
```

报错，错误为

```
ERROR:  学号格式错误!
CONTEXT:  PL/pgSQL function idcheck() line 4 at RAISE
SQL 状态: P0001
```

查看数据，发现依然只有三行；

	id [PK] character varying (20)	name character varying (20)	age integer	sex character varying (20)
1	2011998073	Zhang	19	female
2	2011998088	Zang	20	male
3	2011998099	Luo	20	male

- 建立 `UPDATE` 触发器，对 `student` 表进行 `update` 操作后，若学生学号被修改，则将 `stuexam` 表中相应学号进行修改
  - 输入以下代码，创建触发器函数 `idupdate()`

```
CREATE FUNCTION idupdate()
RETURNS TRIGGER AS $idupdate$
BEGIN
    if (new.id<>old.id) THEN
        UPDATE stuexam set eeid=new.id WHERE eeid=old.id;
    END IF;
    RETURN NEW;
END;
$idupdate$ LANGUAGE plpgsql;
```

  - 创建触发器 `id_update`

```
CREATE TRIGGER id_update
AFTER UPDATE ON student
FOR EACH ROW
EXECUTE PROCEDURE idupdate();
```

  - 先在 `stuexam` 表中插入数据，插入的数据如下

```
INSERT INTO stuexam values('2011998088','database',95);
INSERT INTO stuexam values('2011998073','MachineLearning',94);
INSERT INTO stuexam values('2011998099','DataStruct',92);
```

  - 修改代码如下：

```
UPDATE student set id='211998079' WHERE name='Zhang';
```

  - 查看结果

	id [PK] character varying (20)	name character varying (20)	age integer	sex character varying (20)
1	2011998088	Zang	20	male
2	2011998099	Luo	20	male
3	211998079	Zhang	19	female

	eeid [PK] character varying (20)	major character (20)	grade smallint
1	2011998088	database	95
2	2011998099	DataStruct	92
3	211998079	MachineLearn...	94

- 建立 `DELETE` 触发器， `student` 表中学生记录被删除后，删除 `stuexam` 中该学生相应的考试记录。
  - 建立删除触发器函数`deletestudent()`，代码如下：

```
CREATE FUNCTION deletestudent()
RETURNS TRIGGER AS $deletestudent$
BEGIN
    DELETE FROM stuexam WHERE stuexam.eeid=old.id;
    RETURN NULL;
END;
$deletestudent$ LANGUAGE plpgsql;
```

  - 建立触发器`delete_stu`

```
CREATE TRIGGER delete_stu
AFTER DELETE ON student
FOR EACH ROW
EXECUTE PROCEDURE deletestudent();
```

  - 删除一个记录

```
DELETE FROM student WHERE id='2011998088';
```

	id [PK] character varying (20)	name character varying (20)	age integer	sex character varying (20)
1	2011998099	Luo	20	male
2	211998079	Zhang	19	female

	eeid [PK] character varying (20)	major character (20)	grade smallint
1	2011998099	DataStruct	92
2	211998079	MachineLearn...	94

- 查看所有触发器，删除上面创建的三个触发器及触发器函数。
  - 输入以下代码

```
select * from information_schema.triggers;
```

  结果为

	trigger_catalog name	trigger_schema name	trigger_name name	event_manipulation character varying	event_object_catalog name	event_object_schema name	event_object_table name	action_order integer	action_condition character varying
1	exam_system	public	id_insert	INSERT	exam_system	public	student	1	[null]
2	exam_system	public	delete_stu	DELETE	exam_system	public	student	1	[null]
3	exam_system	public	id_update	UPDATE	exam_system	public	student	1	[null]

  - 删除所有触发器，删除代码为

```
drop trigger id_insert on student;
drop function idcheck();
drop trigger id_update on student
drop function idupdate();
drop trigger delete_stu on student;
drop function deletestudent();
```

实验结果

见实验思路部分

源程序

见实验思路部分

实验体会

触发器可以实现关联表的数据更改，这样可以保证数据的安全性，并且可以进行完整性的约束。

思考题

- 在使用触发器时，对相同的表、相同的事件是否只能创建一个触发器?
  - 输入以下代码，（已存在触发器函数`idcheck()`）

```
CREATE FUNCTION idcheck2()
RETURNS TRIGGER AS $idcheck2$
BEGIN
    IF (CHAR_LENGTH(new.id)<>8) THEN
        RAISE EXCEPTION '学号格式错误! ';
        RETURN NULL;
    END IF;
    RETURN NEW;
END;
$idcheck2$ LANGUAGE plpgsql;
```

  - 触发器

```
CREATE TRIGGER id_insert2
BEFORE INSERT ON student
FOR EACH ROW
EXECUTE PROCEDURE idcheck2();
```

    发现可以继续创建，因此对于相同的表，可以对相同事件创建多个触发器；
- 使用触发器时应注意什么问题?
  - 不能创建具有相同名字的触发器;
  - 触发器有BEFORE触发器和AFTER触发器的区别，执行步骤是先执行 BEFORE触发器，再执行脚本，最后执行AFTER触发器;
  - 不能对本表进行 `insert` , `update` , `delete` 操作,以免递归循环触发
- 为什么要及时删除不再需要的触发器?
  - 防止在重新写触发器程序时涉及到已存在的触发器，导致出现错误;
  - 触发器很消耗资源，会造成程序运行速度慢;
  - 触发器定义之后，每次执行触发事件，都会激活触发器并执行触发器中的语句。如果需求发生变化，而触发器没有进行相应的改变或者删除，则触发器仍然会执行旧的语句，从而会影响新的数据的完整性。因此，要将不再使用的触发器及时删除。