**表的创建**

create table table\_name(column\_name data\_type);--创建普通表

create table student(

id int primary key,

name varchar(32) not null,

age int not null default 0 constraint chk\_age check(age>0),

score int not null

);

insert into student values(1,'zhangsan',12,88);--插入测试数据

insert into student values(2,'lisi',12,78);

insert into student values(3,'wangwu',13,98);

insert into student values(4,'zhaoliu',11,55);

create table table\_name (like source\_table);--复制源表的列名、数据类型以及它们的非空约束。

create table student1 (like student);

create table table\_name (like source\_table including all);--复制源表列上的约束和其他信息

{ INCLUDING | EXCLUDING } { COMMENTS | CONSTRAINTS | DEFAULTS | GENERATED | IDENTITY | INDEXES | STATISTICS | STORAGE | ALL }

create table student2 (like student including all);

create table table\_name as--只复制源表的数据，索引、约束和其他信息都没复制

create table student3 as select \* from student;

create table student4 as table student;

create table orders(

id int,

num int,

price numeric,

total\_price numeric GENERATED ALWAYS AS (num \* price) STORED--生成列

);

insert into orders(id,num,price) values(1,2,99);

PostgreSQL 12 新增特性，之前版本需要函数和触发器才能实现。注意：生成列不能被直接插入或更新，否则会返回错误。

**表的管理**

**查询表**

\d

\dt

\dt+

pg\_class：select \* from pg\_class where relname='tablename';

pg\_tables：select tablename from pg\_tables where schemaname='public';

information\_schema.tables：select table\_name from information\_schema.tables where table\_schema='public';

**查看表结构**

\d table\_name

pg\_attribute：select attname from pg\_attribute where attrelid='tablename'::regclass;

information\_schema.columns： select column\_name from information\_schema.columns where table\_name='t1';

**切换数据库**

\c dbname [username]

\c

\conninfo

**查看表定义语句**

1.pg\_admin

2. pg\_dump -U username -d dbname -s -t tablename -W

-U, --username=名字 以指定的数据库用户联接

-d, --dbname=DBNAME 对数据库 DBNAME备份

-W, --password 强制口令提示 (自动)

-s, --schema-only            只转储模式, 不包括数据

-t, --table=TABLE            只转储指定名称的表

**删除表**

drop table t1;

drop table t2,t3;

**重命名表**

ALTER TABLE [ IF EXISTS ] 名称 RENAME TO 新的名称

**截断表**

truncate t1;

**移动表位置**

ALTER TABLE t1 SET TABLESPACE tbs\_data;--锁表

ALTER TABLE public.t1 SET SCHEMA scm\_test;

**identity自增键的用法**

PostgreSQL中三种自增列serial，identity，sequence，这三个对象从功能性上看，大部分特性都是一致的或者类似的。

三者之间的区别

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 类别 | 是否允许  显示插入 | 显示插入后是否会基于  新的max(id)做自动修改 | truncate table后  是否会自动重置 | 是否跟随事务  回滚一同回滚 | 多对象共享 | 是否支持  手动重置 |
| serial | Y | N | N | N | N | Y |
| sequence | Y | N | N | N | Y | Y |
| identity | Y | N | N | N | N | Y |

sequence示例

create sequence seq\_t1 INCREMENT BY 1 START WITH 1;

create table t1

(

id int not null default nextval('seq\_t1') primary key,

name varchar(10)

);

隐式插入

insert into t1 (name) values ('aaa');

insert into t1 (name) values ('bbb');

insert into t1 (name) values ('ccc');

select \* from t1;

显式插入

insert into t1 (id,name) values (4,'ddd');

select \* from t1;

再次隐式插入

--插入失败，主键重复，因为序列自身是递增的，不会关心表中被显式插入的数据

insert into t1 (name) values ('eee');

--重置序列的最大值

select setval('seq\_t1',(select max(id) from t1)::BIGINT);

insert into t1 (name) values ('eee');--插入正常

--事务回滚后，序列号并不会回滚

select currval('seq\_t1');

begin;

insert into t1 (name) values ('fff');

rollback;

select currval('seq\_t1');

-- truncate 表之后，序列不受影响

truncate table t1;

insert into t1 (name) values ('ggg');

select \* from t1;

--重置序列

ALTER SEQUENCE seq\_t1 RESTART WITH 1;

insert into t1 (name) values ('hhh');

select \* from t1;

serial示例

create table t2

(

id serial primary key,

name varchar(100)

);

insert into t2(name) values ('aaa');

insert into t2(name) values ('bbb');

insert into t2(name) values ('ccc');

select \* from t2;

--显式插入,可以执行

insert into t2(id,name) values (4,'ddd');

--再次隐式插入会报错，因为隐式插入的话，serial会基于显式插入之前的Id做自增，serial无法意识到当前已经存在的最大值

insert into t2(name) values ('fff');

select \* from t2;

--truncate table 后serial不会重置

truncate table t2;

insert into t2(name) values ('ggg');

select \* from t2;

--验证是否会随着事务一起回滚，结论：不会

select currval('t2\_id\_seq');

begin;

insert into t2(name) values ('h');

rollback;

select currval('t2\_id\_seq');

--重置serial，需要注意的是重置的Id必须要大于相关表的字段最大Id，否则会产生重号

SELECT SETVAL('t2\_id\_seq', 1, false);

Identity示例

create table t3

(

id int generated always as identity (cache 100 START WITH 1 INCREMENT BY 1) primary key ,

name varchar(100)

);

insert into t3(name) values ('aaa');

insert into t3(name) values ('bbb');

insert into t3(name) values ('ccc');

select \* from t3;

--显式插入值,如果定义为generated always as identity则不允许显式插入，除非增加overriding system value 提示

--一旦提示了overriding system value，可以

insert into t3(id,name) values (4,'ddd');

insert into t3(id,name)overriding system value values (4,'ddd');

select \* from t3;

--显式插入值,如果定义为generated by default as identity则允许显式插入，

insert into t3(id,name) values (5,'fff');

--显式插入后，继续隐式插入 会报错，identity已经不识别表中显式插入后的最大值

insert into t3(name) values ('ggg');

select \* from t3;

--truncate后再次插入，自增列不会重置

truncate table t3;

select \* from t3;

--truncate并且RESTART IDENTITY后，会重置自增列

TRUNCATE table t3 RESTART IDENTITY;

select \* from t3;

--验证是否会随着事务一起回滚，结论：不会

select currval('t3\_id\_seq');

begin;

insert into t3(name) values ('xxx');

rollback;

select currval('t3\_id\_seq');

--identity自增列的重置表或者更改

ALTER TABLE t3 ALTER COLUMN id RESTART WITH 100;

SELECT setval(t3\_id\_seq, (select max(id) from t3));

**注释**

COMMENT ON TABLE student IS 'This is A Student table'; --表注释

COMMENT ON COLUMN student.id IS 'ID of the student';--列注释

COMMENT ON COLUMN student. name IS 'name of the student';

COMMENT ON COLUMN student.age IS 'age of the student';

COMMENT ON COLUMN student.score IS 'score of the student';

**查看注释**

\d+--查看表注释

\d+ table\_name--查看列注释

查看对象的所有注释

select description from pg\_description join pg\_class on pg\_description.objoid = pg\_class.oid where relname = 'student';

**删除注释**

COMMENT ON TABLE student IS NULL;

**列的管理**

**增加一个新列**

alter table t1 add age int;

alter table t1 add sex boolean;

alter table t1 add score int not null default 60;

alter table t1 add col1 int,add col2 int;

对于表中已经存在的行而言， 最初会给新增的字段填充默认值， 如果没有声明DEFAULT子句， 那么默认填充的是NULL 。

从 PostgreSQL 11开始，添加一个具有常量默认值的列不再意味着在执行ALTER TABLE 语句时需要更新表的每一行。 相反，默认值将在下次访问该行时返回，并在表被重写时应用，从而使得ALTER TABLE即使在大表上也非常快。

**删除列**

alter table t1 drop sex;

在删除某个字段时， 该字段中的内容都会消失， 而且和这个字段相关的约束也会被删除。 如果这个字段被另一个表的外键引用， 删除时就会报错， 如果想删除外键依赖， 需要使用"CASCADE"指明删除所有依赖该字段的内容 。

create table class(id int primary key,name text);

create table student(id int,name varchar,class\_id int references class(id));

alter table class drop id cascade;

**重命名列**

alter table t1 rename name to student\_name;

**修改列的数据类型**

alter table t1 alter col1 type text;

只有字段中现有的项都可以隐式地转换成新类型时， 上面的语句才能执行成功， 否则会执行失败。 示例如下， 把"varchar(40)"改成"varchar(5)"时， 会因为数据放不下而执行失败， 在把"varchar(40)"改成数字时， 会因字符串类型无法隐式转换成数字类型而导致操作失败。

alter table student alter student\_name type varchar(5);--ERROR

改变字段numeric类型的精度， 虽然精度改小后命令执行成功， 但会导致精度数据丢失。

alter table orders alter total\_price type numeric(4,2);--ERROR: numeric field overflow

insert into orders(id,num,price) values(2,3,88.224);--修改成功，但损失精度

alter table orders alter total\_price type numeric(5,2);