

Constraint

A restriction on the values of data allowed within a [table](#), or in attributes of a [domain](#).

primary key, foreign key, not Null, unique, check, default...

Primary Key

A primary key constraint indicates that **a column**, or **group of columns**, can be used as a unique identifier for rows in the table. This requires that the values be **both unique and not null**.

Adding a primary key will automatically create a unique B-tree index on the column or group of columns listed in the primary key, and will force the column(s) to be marked `NOT NULL`.

A table can have **at most one primary key**. (There can be any number of unique and not-null constraints, which are functionally almost the same thing, but only one can be identified as the primary key.) Relational database theory dictates that every table must have a primary key. This rule is not enforced by PostgreSQL, but it is usually best to follow it.

Experiment 1: One way to define Primary Key constraint

Step1: Create a new table emp, let attribution id be the primary key

column_name type primary key

```
create table emp(  
    id int primary key, --Primary Key constraint  
    -- same as UNIQUE NOT NULL  
    name varchar(30),  
    salary numeric(9,2)  
);
```

Step2: You can insert following data manipulation language (DML)

```
insert into emp values(1,'张三',3000);  
insert into emp values(1,'李四',3000);
```

Step3: Insert another DML, watch the result.

```
insert into emp values(1,'王五',3000);
```

Tips: You can create primary key constraint as above format, a new name for this key will also be created at same time automatically by database system.

Experiment 2: Another way to define Primary Key constraint

PRIMARY KEY (column_name)

or you can **name this primary key yourself**

constraint pk_name PRIMARY KEY (column_name)

Step1: Create a table emp2, let attribution id be the primary key

```
create table emp2(  
  id int,  
  name varchar(30),  
  salary numeric(9,2),  
  constraint pk_emp2 primary key(id)  
);
```

Step2: Insert following records

```
insert into emp2 values(1, '张三', 3000);  
insert into emp2 values(2, '李四', 3000);
```

Step3: Insert another record, watch the result

```
insert into emp2 values(1, '李四', 3000);
```

Experiment 3: Define Union Primary Key constraint

PRIMARY KEY (column_name1, column_name2, ...)

Step1: Create a table emp_union, let attribution id be the primary key

```
create table emp_union(  
  name varchar(30),  
  dep varchar(20),  
  salary numeric(9,2),  
  primary key(name, dep)  
);
```

Step2: Insert following records

```
insert into emp_union values ('张三', '研发', 5000),  
                             ('李四', '人事', 3000);
```

Step2: Test following DML, watch the result

```
insert into emp_union values ('张三', '财务', 3000);--yeah  
insert into emp_union values ('张三', '研发', 3000);--wow
```

Experiment 4: Primary Key VS UNIQUE NOT NULL

Step1: Create follow tables, and contrast the results

```
create table emp_pk(  
  id int primary key ,  
  name varchar(30) primary key ,  
  salary numeric(9,2)  
);--wow  
  
create table emp_unu(  
  id int unique not null ,  
  name varchar(30) unique not null,  
  salary numeric(9,2)  
);--yeah  
  
create table emp_pk_unu(  
  id int primary key ,  
  name varchar(30) unique not null,  
  salary numeric(9,2)  
);--yeah
```

Tips: There can be any number of unique and not-null constraints, which are functionally almost the same thing, but only one can be identified as the primary key.

Foreign Key

A foreign key constraint specifies that the values in **a column (or a group of columns)** must match the values appearing in some row of **another table**.

Experiment 5: One way to define Foreign Key constraint

```
column_name type REFERENCES another_table(column_name2)
```

Step1: Create the "another" table, and add some records

```
create table dept(  
  id int primary key,  
  name varchar(40)  
);  
  
insert into dept values(1, '开发部');  
insert into dept values(2, '测试部');
```

Step2: Create a table with a foreign key

```
create table emp5(  
  id int primary key,  
  name varchar(30),  
  salary numeric(9,2),  
  deptId int REFERENCES dept(id)  
);
```

Step3: Insert some records, watch how foreign key works

```
insert into emp5 values(1, '张三', 3000, 1);
insert into emp5 values(2, '李四', 3000, 3);
```

Tips: You can create foreign key constraint as above format, a new name for this key will also be created at same time automatically by database system.

Experiment 6: Another way to define Foreign Key constraint

constraint fk_name foreign key(column_name1) REFERENCES another_table(column_name2)

Step1: Create a table with foreign key also reference to dept table column id

```
create table emp6(
    id int primary key,
    name varchar(30),
    salary numeric(9,2),
    deptId int,
    constraint fk_dept FOREIGN KEY(deptId) references dept(id)
);
```

Step2: Add following records, and watch the result

```
insert into emp6 values(1, '张三', 3000, 1);
insert into emp6 values(2, '李四', 3000, 3);
```

Tips: You can name foreign key yourself in this create method.

Tips: A foreign key can also constrain and reference a group of columns. As usual, it then needs to be written in table constraint form. Here is a contrived syntax example

```
CREATE TABLE t1 (
    a integer PRIMARY KEY,
    b integer,
    c integer,
    FOREIGN KEY (b, c) REFERENCES other_table (c1, c2)
);
```

Tips: Primary key/foreign key is a kind of index, could improve efficiency of query

NOT NULL, UNIQUE, DEFAULT, CHECK

A not-null constraint simply specifies that a column must not assume the null value.

Unique constraints ensure that the data contained in **a column**, or **a group of columns**, is unique among all the rows in the table.

A check constraint is the most generic constraint type. It allows you to specify that the value in a certain column must satisfy a **Boolean (truth-value) expression**.

Experiment 7: NOT NULL constraint

Step1: Create a new table with NOT NULL constraint

```
create table emp7(  
  id int primary key,  
  name varchar(30) not null,  
  salary numeric(9,2)  
);
```

Step2: Add some records, and watch the result with "select * from ..."

```
insert into emp7 values(1, '张三', 3000);  
insert into emp7 values(2, null, 3000);  
select * from emp7;
```

Experiment 8: UNIQUE constraint

Step1: Create a table with UNIQUE constraint and NOT NULL constraint.

```
create table emp8(  
  id int primary key,  
  name varchar(30) not null,  
  phone varchar(30) unique,  
  salary numeric(9, 2)  
);
```

Step2: Add some records, watch the contrast of result

```
insert into emp8 values(1, '张三', '13611111111', 3000);  
insert into emp8 values(2, '李四', '13611111111', 3000);  
insert into emp8 values(3, '王五', null, 3000);  
insert into emp8 values(4, '罗六', null, 3000);  
insert into emp8 values(null, '庄七', '1362222222', 3000);  
select * from emp8;
```

Tips: For UNIQUE constraint, you can insert NULL value, what's more, many NULL values; but for primary key, NULL value is illegal; here can be many UNIQUE constraints, but only one Primary key.

Tips: The following syntax is also legal.

```
CREATE TABLE products (  
  product_no integer,  
  name text,  
  price numeric,  
  UNIQUE (product_no)  
);
```

```
CREATE TABLE example (  
  a integer,  
  b integer,  
  c integer,  
  UNIQUE (a, c)  
);
```

Experiment 9: CHECK constraint

Step1: Create a table with check constraint

```
create table emp9(  
  id int primary key,  
  name varchar(30) ,  
  phone varchar(30) ,  
  salary numeric(9,2) CHECK ( salary>0 )  
);
```

Step2: Add some records, and watch the result.

```
insert into emp9 values(1, '张三', '13611111111', 3000);  
insert into emp9 values(2, '李四', '13611111111', -3000);
```

Tips: You can also name a check constraint a separate name using following syntax

```
CREATE TABLE products (  
  product_no integer,  
  name text,  
  price numeric CONSTRAINT positive_price CHECK (price > 0 and price<=100000)  
);
```

Tips: A check constraint can also refer to several columns.

```
CREATE TABLE products (  
  product_no integer,  
  name text,  
  price numeric CHECK (price > 0),  
  discounted_price numeric CHECK (discounted_price > 0),  
  CHECK (price > discounted_price)  
);
```

Experiment 10: DEFAULT constraint

Step1: Create a table with check constraint

```
create table emp10(  
  id int primary key,  
  name varchar(30) not null,  
  salary numeric(9,2) default 0.0  
);
```

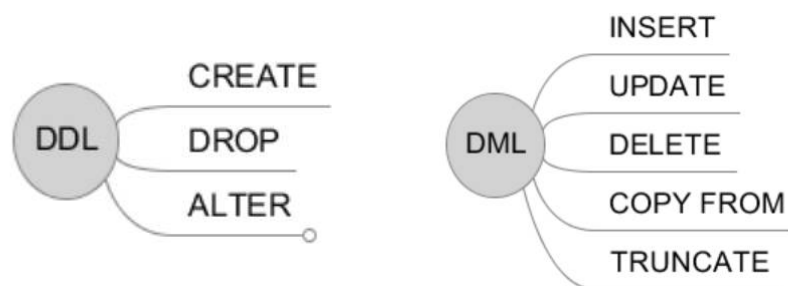
Step2: Add some records, and watch the result.

```
insert into emp10(id, name) values(1,'张三');  
insert into emp10(id, name, salary) values(2,'李四',3000);  
select * from emp10;
```

DDL vs DML

DDL: data definition language

DML: Data Manipulation Language



DML

Data manipulation language: **insert, update, delete, truncate**

Using the following table

```
create table emp10(  
  
  id int primary key,  
  
  name varchar(30) not null,  
  
  salary numeric(9,2) default 0.0  
  
);  
  
insert into emp10(id, name) values(1,'张三');  
  
insert into emp10(id, name, salary) values(2,'李四',3000);  
  
select * from emp10;  
  
drop table emp;
```

```
drop table dept;

create table dept(

id int primary key,

name varchar(40)

);

create table emp(

id int primary key,

name varchar(30),

salary numeric(9,2),

deptId int references dept(id)
```

Experiment 11: Insert— create new rows in a table

INSERT INTO table(...) VALUES(..);

<https://www.postgresql.org/docs/current/sql-insert.html>

Tips: When inserting a lot of data at the same time, consider using the COPY command. It is not as flexible as the INSERT command, but is more efficient.

Experiment 12: Update— update rows of a table

UPDATE table SET ...;

<https://www.postgresql.org/docs/current/sql-update.html>

```
UPDATE dept SET name='市场部' where id=4;
```

Tips: To update existing rows, use the UPDATE command. This requires three pieces of information:

1. The name of the table and column to update
2. The new value of the column
3. Which row(s) to update

Experiment 13: Delete— delete rows of a table

DELETE FROM table WHERE ...;

<https://www.postgresql.org/docs/current/sql-delete.html>


```
DELETE FROM dept WHERE name='市场部';
```

```
DELETE FROM dept;
```

Tips: You can also remove groups of rows matching a condition, or you can remove all rows in the table at once.

Experiment 14

Step1:Create a table with table head and suitable constraint as follow

sid	name	gender	age	addr
cs307_lab5				
sid	numeric			
name	varchar(30)			
gender	varchar(5)			
age	integer			
addr	varchar(50)			
pk_stu	(sid)			
pk_stu	(sid) UNIQUE			
cs307_lab5_age_check	(age > 1)			

Step2:INSERT some records into the table created in step1, ;

Step3: UPDATE some data in the table

Step4:DELETE one record

Step5:TRUNCATE the whole table

example:

```
CREATE table cs307_lab5
(
    sid    numeric,
    name   varchar(30),
    gender varchar(5),
    age    int check ( age > 1 ),
    addr   varchar(50),
    constraint pk_stu primary key (sid)
);

insert into cs307_lab5
values (1, 'zhangsan', 'M', 19, 'department 9'),
       (3, 'lisi', 'F', 18, 'joy land');

UPDATE cs307_lab5 set age=19 where name='lisi';

DELETE from cs307_lab5 where name='lisi';

TRUNCATE cs307_lab5;
```

