1. **PostGreSQL** **Basics:**

**Integer type:**

smallint **// Small range integer, value range:**-32768~32767；

int(integer) **// Normal size integer** -2147483648~2147483647；

**Arbitrary precision floating point number type：**

real **//6-bit decimal digital precision;**

numeric(m,n) **//Any precision type, m is the maximum length of the value, n is the reserved decimal, and the decimal point is not occupied；**

**Date and time type:**

time **// Only for the time of day，example**  10:05:05；

date **// Used only on dates， example**  1987-04-04；

timestamp **// Date and time， example**  1987-04-04 10:05:05；

**String type:**

char(n)/character(n)  **// Fixed length n string, insufficient to fill in blanks;**

varchar(n)/character varying(n) **// Variable length string with length limit；**

text  **// Variable length string with no length limit**; **Common numeric functions:**

avg() **// Returns the average of a column;**

count() **// Returns the number of rows in a column；**

max() **// Returns the maximum value of a column；**

example：select max(e\_salary) from employee;// Query the maximum wage；

min() **// Returns the minimum value of a column**

example：select min(e\_salary) from employee;/**/ Query the minimum wage；**

sum() **// Returns the sum of the values of a column；**

**Commonly used string functions：**

length(s) **// Calculate the length of a string；**

example：select e\_name,length(e\_name) from employee;

**// Query employee name and its name length；**

concat(s1,s2,...) **// String merging function** ；

example：select e\_no, e\_name, e\_hireDate, concat(c\_no, e\_name, e\_hireDate) from employee;

ltrim(s)/rtrim(s)/trim(s) **// Delete string space function；**

replace(s,s1,s2) **// String substitution function；**

substring(s,n,len) **// Get substring function；**

**Syntax for creating a function：**

creat function **// Declare the creation function；**

add(integer,integer) **// Define function name, parameter type；**

return integer **// Define the return value of the function；**

as 'select $1 + $2;' /**/ Define the function body；**

language SQL **// Create functions through SQL language；**

returns NULL ON NULL INPUT; **// Handling when the parameter is defined as NULL；**

**Operate the database：**

1、Create a database object：create database Database name;

create database db\_jikexueyuan # Database name；

with ENCODING='utf-8' # Code as utf-8；

OWNER=user # Owner；

CONNECTION LIMIT=10; # Limit the number of connections to 10；

2、Modify database objects (some of these parameters)：alter database ...

(1)alter database db\_jikexueyuan

rename to db\_jikexueyuan1; # Modify database name；

(2)alter database db\_jikexueyuan1

with CONNECTION LIMIT = 100; # Modify database connection limit；

or alter database db\_jikexueyuan1 CONNECTION LIMIT 100; It's the same；

Table-level operation：

(3)alter table student rename to student1;

# Modify table name: Modify table name student to student1；

(4)alter table student1 rename id to bh;

# Modify field name: Modify the field id of table student1 to bh；

(5)alter table student1 alter column name type varchar(40);

# Modify the field data type. Modify the data type of the name field to VARCHAR (40)；

(6)alter table student1 drop column birthday;

# Delete the field. Delete the field birthday, and delete the corresponding information；

(7)alter table student1 add column address varchar(200);

#Add a field. Add an address field of type VARCHAR(200)；

3、Delete database objects: drop database database name;

Table-level operation：

(1)drop table student1; # Delete data tabl;

(2)drop table if exists student1; # Delete is performed only if the datasheet exists;

4、Manipulate datasheet objects：create table table\_name ...

create table student (

id int,

name varchar(30),

birthday date,

score numeric(5,2)

);

5、Update database tables：

(1) Specify conditional update data: update table name set field name to update = field content to update where which data to update;

update student set name = 'lisi' where id = 2; // Update the student name of id=2 to "lisi"

(2) Batch update data

update student set score = 0; // Update all credits in the student table to 0

insert into student (id, name, birthday) values (2, 'zhangsan', '1990-01-02');

// The part that is not inserted is null

insert into student (id, name, birthday) values

(3, 'zhangsan1', '1990-02-01'),

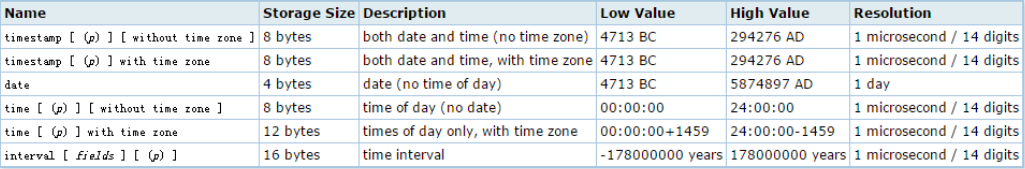
(4, 'zhangsan2', '1990-02-02'), // Insert data in bulk

1. **SQL server与PostgreSQL对比：**

1、SQLserver: Date: is the date type, the same as the date type in PostgreSQL

2、SQLserver: Datetime: is the date type (plus time), which is 8 bytes as the datestamp type in PostgreSQL, but datestamp makes a detailed distinction in the time zone, and there is no datetime type in PostgreSQL

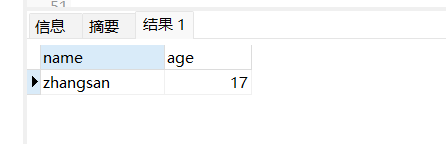
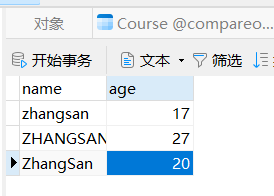
3、The time type of PostgreSQL is as follows, and can be referenced by PostgreSQL：[PostgreSQL 日期/时间类型 (yiibai.com)](https://www.yiibai.com/manual/postgresql/datatype-datetime.html)；



4、Postgresql does not support nvarchar type ……

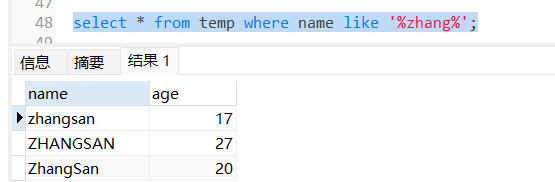
5、Table data：

Same function：select \* from temp where name like '%zhang%';



**Database table 1**

**Figure 1 PostgreSQL query results**



**Figure 2 SQL server query results**

Conclusion：

PostgreSQL is case-sensitive when blurring queries; it is not case-sensitive in the following SQLserver:

When PostgreSQL and SQL Server's fuzzy matching like is different, PostgreSQL's like is case-sensitive and SQL Server is not case-sensitive. If you want PostgreSQL's like to be case-insensitive, you can use ilike. Or use lower or upper to convert to lowercase or uppercase and then fuzzy matching. In this way, SQL is compatible with both databases.

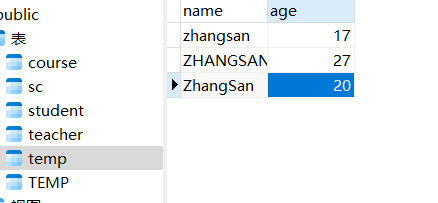
select \* from t\_user where name ilike '%zhang%';

select \* from t\_user where upper(name) like upper('%zhang%');

select \* from t\_user where lower(name) like lower('%zhang%');

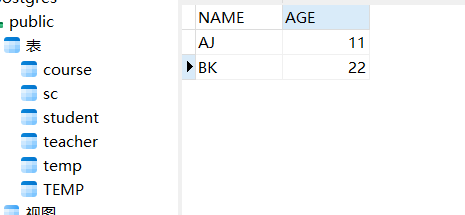
6、 PostgreSQLTable and field names should be in double quotation marks or they will become lowercase characters.

Execute PostgreSQL statement*:*select \* from TEMP……



**Figure 3 PostgreSQL execution result**

Execute again PostgreSQL*：*select \* from " TEMP" ……



**Figure 4 implementation result of PostgreSQL with quotation marks**

Conclusion：

PostgreSQLWhen operating the database, table and field names should be enclosed in quotation marks. If you do not add them, you will not be case-sensitive.

Solution：

Table name： SQLSERVER-> DCS\_RISK == POSTGRESQL->”DCS\_RISK”;

Filed name： SQLSERVER-> STATUS\_NAME == POSTGRESQL->”STATUS\_NAME”

Remark ：（In the sqlserver table, adding "double quotation marks" to table and field names will not report errors and there is no "character case" control）.

7、The ISNULL function in Sqlserver cannot be used in postgresql!

Solution：ISNULL(IS\_CLOSED,0)=0 == coalesce("IS\_CLOSED" IS NULL OR "IS\_CLOSED"=0)

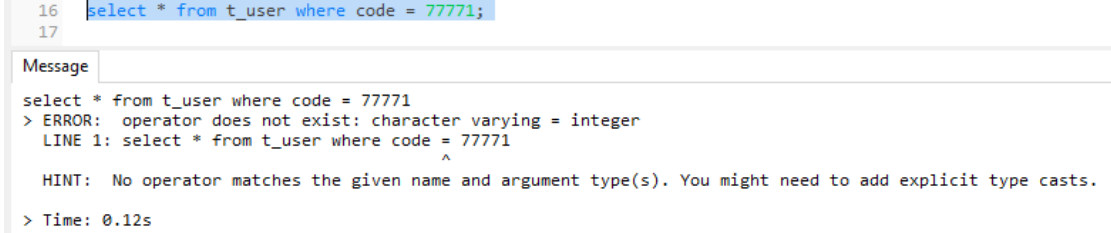
8、In Sqlserver, the character type is numeric, and querying by empty string can be used. In postgresql, the character type is numeric is not available!

example：In PostgreSQL:

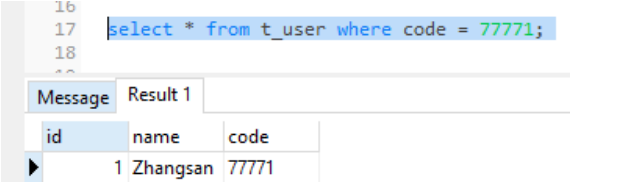
The AGE in the TEMP table is of type int. An error will be reported if you use AGE= "" as a condition!

9、PostgreSQL is strongly typed when doing conditional queries, but SQL Server is weakly typed. In other words, using numeric type values, querying character type values report errors and need to be enclosed in single quotation marks, while querying array type values with string values does not report errors.

Execute statement：select \* from t\_user where code = 77771;



**Figure 5 PostgreSQL execution result**

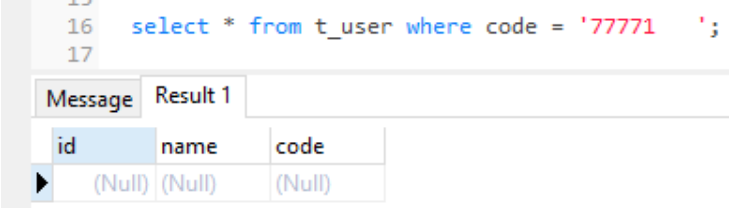


**Figure 6 SQLserver execution result**

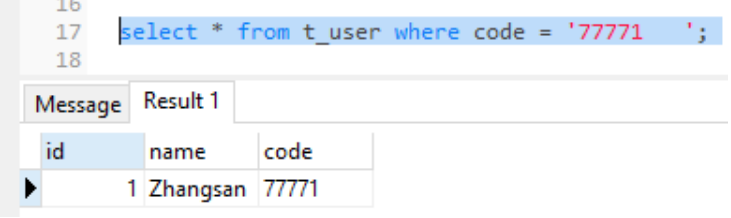
Conclusion：

// Code is a character type 77771 is a data value type, the execution will report an error, so the query should be enclosed in '77771' single quotation marks

10、If there is a blank at the end of SQL Server's query, SQL Server will ignore it, but PostgreSQL will not.

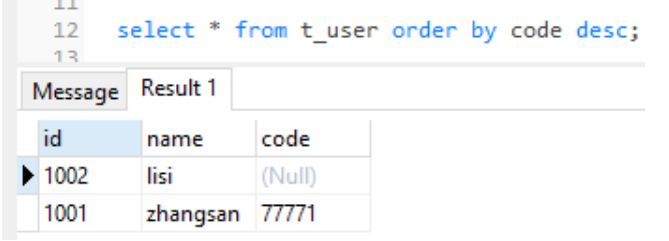


**Figure 7 PostgreSQL execution result**

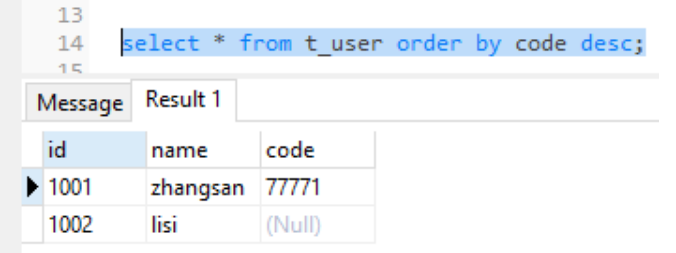


**Figure 8 SQLserver execution result**

11、The default order by behavior of PostgreSQL and SQL Server is inconsistent. If the field of order by is null,PostgreSQL, it will be put in front of it, and SQLServer will put it after it.



**Figure 9 PostgreSQL-order by**



**Figure 10 SQLserver-order by**

Solution：

In some cases, if the order of the data is required to be consistent between the two databases, you can add nulls last to the query SQL of PostgreSQL to make the null data lag.

example：select \* from t\_user order by code desc nulls last;

You can also use case when to unify SQL：

example：ORDER BY (case when xxx is null then '' else xxx end) DESC;

12、String concatenation：

SQL Server Use the "+" sign to concatenate strings, and the concat function is not supported before version 2012. PostgreSQL uses "| |" to concatenate strings and supports the concat function.

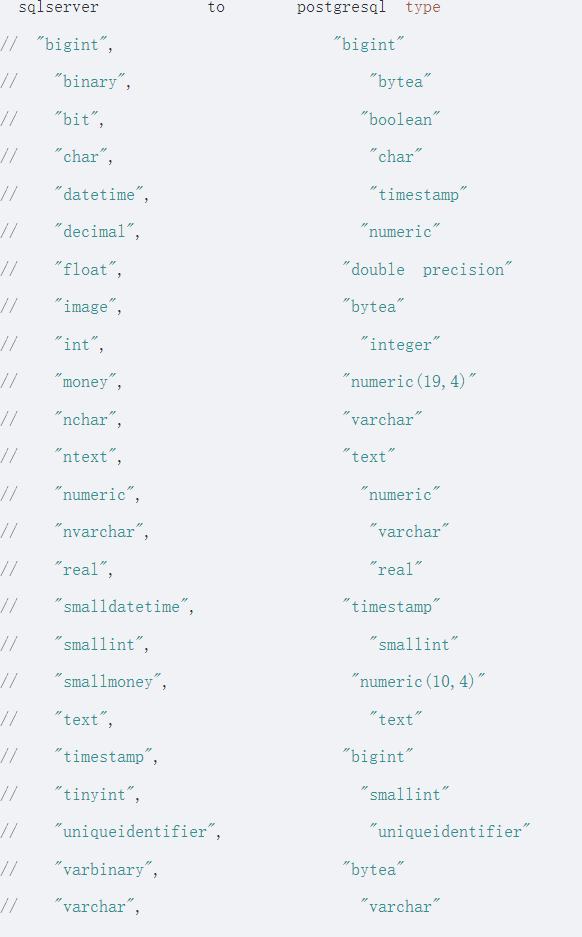
13、Query whether the table exists：

//SQL Server

select count(name) from sys.tables where type='u' and name='t\_user';

//PostgreSQL

select count(table\_name) from information\_schema.tables where table\_name='t\_user';



**Figure 11 comparison table of field types between SqlServer and Postgresql database**