1. IDE 3

Navicat的使用 3

基本命令 3

2.1 基本 3

2.1.1 修改表结构 3

2.1.2 Join & union 3

2.1.3 关键字 3

2.1.4 数据结构/类型 3

2.1.4.1 json |jsonb |string 3

2.1.4.2 GIN indexing 4

2.1.4.3 B-Tree 4

2.1.4.x 实际的问题 4

2.2 命令行操作 5

2.2.1 命令行登陆 5

2.2.2 命令行备份(backup) 5

2.2.3创建用户 5

2.3 库函数 5

2.3.1 substring(str, start[, end]) 5

2.3.2 to\_number(text,text) 5

2.3.3 lower(str) 6

2.3.4 replace 6

2.3.5 trim 6

2.3.6 length | char\_length | bit\_length 6

2.4 Aggregate 6

2.5 foreign key 6

3. 存储程序 6

json写入数据表中 6

4. 事务处理 6

5. 部署问题 7

lock: https://www.postgresql.org/docs/10/static/monitoring-stats.html

常用sql语句

select pc.name as country, pv.name,pv."enName",pv.ctrip\_code from place.cities as pc

LEFT JOIN place.cityview as pv

on pc.id = pv."parentId" and pc.type =2 and pc."isAbroad" =true;

with recursive r as (

select \* from place.cities as pc where type =2 and "isAbroad" =TRUE

union all

select pv.\* from place.cities as pv , r where pv."parentId" =r.id

)

select name, "parentId" from r order by id;

create table data as (select pc.name as country, pv.name,pv."enName",pc.ctrip\_code from place.cities as pc

LEFT JOIN place.city2 as pv

on pc.id = pv."parentId" and pc.type =2 and pc."isAbroad" =true);

create table datas as (select pc.name as country, pv.name,pv."enName",pv.ctrip\_code from place.cities as pc

LEFT JOIN place.cityview as pv

on pv."parentId" = pc.id and pv.name is not null);

获取schema：

select table\_schema from information\_schema.tables;

# 备份和复原

## 1. 备份说明

备份：整个数据库、某些schema、某些table等，备份必备参数: database\_name, 对于--前缀提供的参数，注意不能存在多余空格

对于—指定的参数，=后可加引号，也可不加

### 1.1 备份数据库

pg\_dump --file "times1.sql" --encoding "utf-8" -U jack --format=tar -h localhost -p 5432 times

### 1.2备份某个schema下的数据库

-n schema

--schema=schema\_name

上述两种写法均可

pg\_dump --file "times1.sql" -n approve --encoding "utf-8" -U jack --format=tar -h localhost -p 5432 times

以往记录

pg\_dump --file "times2.sql" --schema=approve --encoding "utf-8" -U jack --format=tar -h localhost -p 5432 times

pg\_dump --file "attachment.sql" --schema=attachment --encoding "utf-8" -U jingli --format=tar -h localhost -p 5432 api

## 2. 还原说明

还原：整个数据库、某些schema、某些table等，还原必备参数: 指定还原目标文件的文件名，对于--前缀提供的参数，注意不能存在多余空格

### -e 参数表明遇到错误，停止执行， 无需在其后添加参数

### --filename 复原的该参数用来将执行复原的log写入文件，而非z

### 2.1 还原数据库

pg\_restore attachment.sql -h localhost -p 5432 -U jack --dbname='for\_test' --format=tar

### 2.2 还原数据库下某个schema

pg\_restore times1.sql -h localhost -p 5432 -U jack --dbname='for\_test' --format=tar --schema=approve --table=approves

以往记录

pg\_restore times1.sql -h localhost -p 5432 -U jack --dbname='for\_test' --format=tar --schema='approve' --table='approves'

pg\_restore -h localhost -p 5432 -U jack --dbname='for\_test' --file =times1.sql --format=tar --schema='approve' --table='approves'

pg\_restore -h localhost -p 5432 -U jack -d for\_test -e -f './times1.sql' --format=tar --schema=approve --table=approves

pg\_restore times1.sql -h localhost -p 5432 -U jack --dbname='for\_test' --schema=approve --table=approves

pg\_restore attachment1.sql -h localhost -p 5432 -U jingli --dbname='qmtrip' --schema=attachment --table=attachments

## 3. window系统

使用windows的cmd命令进行数据库复原操作

D:\postgresql\bin\pg\_dump.exe --host "l.jingli365.com" --port "15432" --username "times" --dbname "test\_time" --verbose --section pre-data --section data --section post-data " -f “D:\backup\test\online\test\_times.sql" --T “travel\_budget\_logs”

D:\postgresql\bin\pg\_dump.exe --host "l.jingli365.com" --port "15432" --username "times" --verbose –-format=c –-encoding “utf-8” –-file “D:\backup\test\online\test\_times.sql” --exclude-table “travel\_budget\_logs” “time\_test”

参考： https://www.postgresql.org/docs/9.2/static/app-pgrestore.html

正确：

D:\postgresql\bin\pg\_dump.exe --host "l.jingli365.com" --port "15432" --username "times" --verbose --format=c --encoding "utf-8" --file "D:\backup\test\openapi.sql" "openapi"

D:\postgresql\bin\pg\_dump.exe --host "j.jingli365.com" --port "5432" --username "jingli" --verbose --format=c --encoding "utf-8" --file "D:\backup\temp\qmtrip.sql" --exclude-table "travelbudget.travel\_budget\_logs" "qmtrip"

D:\postgresql\bin\pg\_restore.exe --host "127.0.0.1" --port "5432" --username "postgres" --verbose --format=c --encoding "utf-8" --file "D:\backup\test\jlbudget.sql" -d "jlbudget"

# 1. IDE

## Navicat的使用

ctrl+q 打开查询窗口

ctrl+/ 注释

ctrl +shift+/ 解除注释

ctrl+r 运行选中的SQL语句

ctrl+l 删除选中行内容

ctrl+d 复制当前行内容并粘贴到下一行

ctrl+w 关闭当前查询窗口

# 基本命令

查看软件版本

# 访问不对外开放端口的数据库（ssh）

https://www.postgresql.org/docs/9.1/static/ssh-tunnels.html

# 2.1 基本

$ilike

左连接：

SELECT \* FROM auth.tokens as at LEFT JOIN auth.accounts as aa on at.id=aa.id;

使用联合查询来更新多张表

### 2.1.1 修改表结构

#### 2.1.1.1 添加字段

alter table place.cities add column euroRailCtripCode varchar(10);

#### 2.1.1.2 删除字段

alter table place.cities drop column euroRailCtripCode [CASCADE];

CASCADE: optional，to remove all constraints which is relevant to this column.

#### 2.1.1.3 修改字段类型

alter table place.cities alter column euroRailCtripCode type varchar(10);

type: keyword in postgres

alter table statistics.memory alter column id type varchar(100);

#### 2.1.1.4 修改/删除字段默认值

alter table staff.staffs alter column price set DEFAULT 7.77;

alter table staff.staffs alter column price drop DEFAULT;

SET、DEFAULT: keywords in postgres

#### 2.1.1.5 重置字段名

alter table staff.staffs RENAME column price TO new\_price;

RENAME、TO: keyword in postgres

#### 2.1.1.6 添加/删除约束

alter table staff.staffs alter column name SET NOT NULL;

alter table staff.staffs drop constraint some\_name; //known constraint’s name

alter table staff.staffs alter column name DROP NOT NULL; //unknown constraint’s name

SET、NOT NULL: keywords in postgres

### 2.1.2 Join & union

Union是将多个result set进行联合。一般union两段为select clause。同时需要注意：多个select clause必须满足相同数量和类型的column并以相同顺序显示。其中column的长度可以不同。

### 2.1.3 关键字

Only

当查询时，from clause后、表名前可添加only，表示只在当前表中进行查询，对于其inherit 的子表，不进行查询。

Inherit

创建表时可以使用

### 2.1.4 数据结构/类型

#### 2.1.4.1 json |jsonb |string

参考：<https://stackoverflow.com/questions/22654170/explanation-of-jsonb-introduced-by-postgresql>

<https://www.postgresql.org/message-id/E1WRpmB-0002et-MT@gemulon.postgresql.org>

JSONB 存储时，需要进行转换编码，较为耗时，其多数情况其存储占较大空间。但是读取快速，无需re-parsing. 可以进行aggregate查询。Jsonb包含除json含有的JSON generation functions(to\_json, json\_agg等）函数以外的所有函数和操作符。 此外， jsonb含有operator class，用于hash和btree 索引。同时有GIN 索引的类。

Json：以一种text形式进行存储，存储迅速，读取较慢。

HStore 文档类型数据的可选列类型。在jsonb之前，已被jsonb取代。

**弊端：**

Jsonb类型并不保存不必要的whitespace、duplicate keys、key ordering。 Jsonb数据保存最后一个key值。

**Operation class**

参考：<http://schinckel.net/2014/05/25/querying-json-in-postgres/>

？：表示查询json中存在某个字段。

？|：表示查询json中是存在列表中的任意一个key

？&：表示查询json中存在列表中的所有keys

**->**：表示查询json中的某个key的值

**->>**：

**#>：类似嵌套分层级进行查询, 查询json中一级数组中的某项的值**

**Usage:**

select budget\_config ->'hotel' as hotel from company.companies; //budget\_config 结构为{“hotel”:{}, “trafffic”:{}}

select \* from company.companies where auto\_approve\_preference @>'{"beforeDeparture":"5h"}';

select \* from company.companies where auto\_approve\_preference ? 'beforeDeparture';

select budget\_config::json #> ‘{traffic, 2}’ from budget.budget\_items; //budget\_config结构为： {“hotel”:{}, “trafffic”:[{}, {}, {}]}

#### 2.1.4.2 GIN indexing

GIN indexing：

Generalized inverted index。使用btree.

#### 2.1.4.3 B-Tree

Self-balancing tree data structure.是external memory的很好的data structure。通常用于databases 和filesystem。非子节点拥有可变数量的子节点。一个节点的分支（或子节点）的数量会比存储在节点内部键值的数量大1

#### 2.1.4.4 json格式问题

对于类型为jsonb的数据，在insert或者update数据时，需要注意格式

update travel\_policy.company\_regions set types = '[1,2,3]', "group" = 2 where name = '港澳台'; // 字段types为jsonb类型

json数据插入时，\不会被视为转义字符，当且仅当前置E时。当数据中存在单引号时，如jack’s name此时的单引号需要转义，postgres支持如下方法：

---使用成对的单引号实现转移： insert into trip\_plan.trip\_approves(id, query) values(‘00000’, ‘jack’’s name’);

---对于插入字段的引号前加入E,表明是posix(Portable Operating System Interface of UNIX)的编码. 会对含有backslash的值，会将其视为转移字符

insert into trip\_plan.trip\_approves(id, query) values(‘00000’, E‘jack\’s name’);

#### 2.1.4.5 数字类类型

##### 2.1.4.5.1 int-整型

###### 2.1.4.5.1.1 int

2个字节

###### 2.1.4.5.1.2 integer

4个字节

###### 2.1.4.5.1.3 bigint

8个字节(二进制表示2^64), 范围： -9223372036854775808 to +9223372036854775807（十进制表示10^18）

##### 2.1.4.5.2 float-浮点型

###### 2.1.4.5.2.1 decimal

user-specified precision. Exact.

Up to 131072 before decimal point, up to 16383 after decimal point. Which means, almost 17～18 bit before decimal point, 14 bit after decimal point

For example:

23.5141 has the precision of 6 and scale of 4

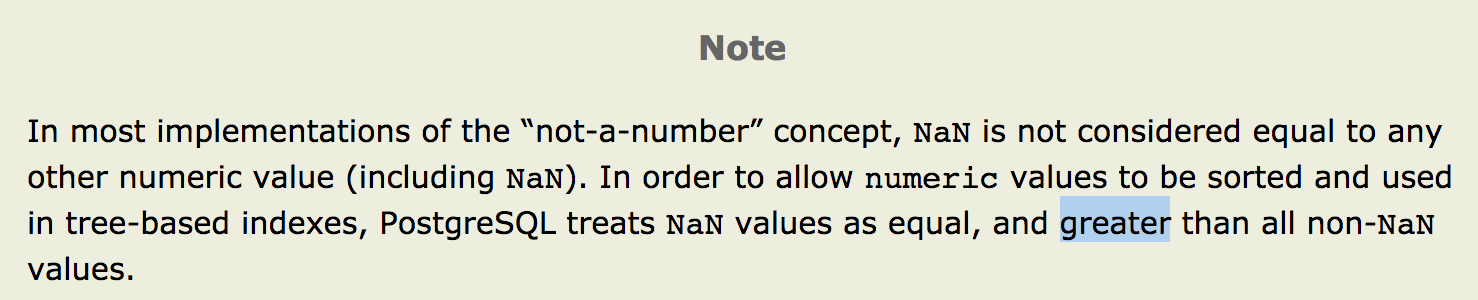
###### 2.1.4.5.2.1 numeric

user-specified precision. Exact

Up to 131072 before decimal point, up to 16383 after decimal point. Which means, almost 17～18 bit before decimal point, 14 bit after decimal point

***Note:***

Type numeric allow special value: NaN (not a number) to be stored. E.g, update table set a = ‘NaN’



###### 2.1.4.5.2.1 real

4 bytes, Inexact

###### 2.1.4.5.2.1 double precision

8 bytes, Inexact.

###### 2.1.4.5.2.1 smallserial

2 bytes, autoincrementing integer, only positive integer

###### 2.1.4.5.2.1 serial

4 bytes, autoincrementing integer, only positive integer

Range: 1 to 2147483647

###### 2.1.4.5.2.1 bigserial

8 bytes, autoincrementing integer, only positive integer

Range: 1 to 9223372036854775807

#### 2.1.4.6 文本类型

###### 2.1.4.6.1 string

###### 2.1.4.6.2 text

#### 2.1.4.7 uuid

uuid类型由32digits(该32个digits是使用十六进制表示)组成，128个bit，共占用16个字节。每个digit占用4bit

### 2.1.5 多表联合操作

#### 2.1.5.1 多表联合查询

#### 2.1.5.2 多表联合更新

方法一：使用from的方式(缺陷：每次只能更新一张表)

update approve.approves as aa

set budget = tt.budget, old\_budget = tt.old\_budget

from trip\_plan.trip\_approves as tt

where aa.id = tt.id

注意，update语句中的set其后跟的表字段，无需使用别名进行约束，更新字段为update关键字后的表名。

方法二：

通过join的方式进行表的链接，进而根据条件进行相应的字段的更新

方法三

使用with， 相当于创建视图，将修改反射到原表

## 2.2 命令行操作

reference: <http://postgresguide.com/utilities/psql.html>

### 2.2.1 命令行登陆

远程登录： Psql -h ip -d database -U

#### 2.2.1.2 重启数据库服务

/etc/init.d/postgresql restart

### 2.2.2 命令行备份(backup)

postgresql提供备份工具：pg\_dump

使用举例：

pg\_dump --file 存储文件目录 --host ip\_address --port port\_number --username userName --no-password --verbose --role “admin” --format=c --blobs --section=pre-data --section= post-data --encoding “utf-8” ***“times”***

对于需要登陆的数据库，备份时需要将免密参数信息去除； 命令的最后需要指定数据库名称”times”

链接数据后，

---\d tablesname：用于显示表结构。E.g: \d task.tasks (先引用schema.tablename)

### 2.2.3用户/数据库/schema/表

#### 2.2.3.1 查询语句

显示角色信息： \dg [S+] [PATTERN] 或者 \du [S+] [PATTERN]

#### 2.2.3.2 创建语句

CREATE DATABASE times\_common\_api WITH OWNER times;

CREATE USER user-name WITH PASSWORD password

CREATE USER times WITH PASSWORD ‘time0418’;

CREATE DATABASE times WITH OWNER times;

CREATE SCHEMA name; (通过\connect进行数据库切换，进而可以创建某个数据库的schema);

CREATE SCHEMA IF NOT EXISTS schema\_name AUTHORIZATION user\_name;

CREATE TABLE statistics.memory(id INT primary key, heapusage DECIMAL(10,3), heaptotal DECIMAL(10,3)); //命令行登陆的owner是谁，默认创建的表的owner就是谁

### 2.2.4 快捷键（显示用户列表）

\du 显示用户列表

\c 显示链接到的数据库

\d 显示当前数据库表格 \d table-name 显示该表的结构

### 2.2.5 创建数据库

CREATE DATABASE times OWNER postgres;

CREATE DATABASE times\_common\_api WITH OWNER times; //添加with关键字，成功

### 2.2.6 授权

GRANT ALL PRIVILEGES ON DATABASE exampledb to dbuser;

### 2.2.7 删除数据库/schema

DROP DATABASE times;

DROP SCHEMA IF EXISTS schema\_name CASCADE; //级联删除schema，及其内部的table、function

### 2.2.8 数据库内切换数据库

\connect database\_name;

### 2.2.9 显示数据库的系统级信息（schema、funciton等）

参考： <https://gist.github.com/Kartones/dd3ff5ec5ea238d4c546>

#### 2.2.9.1 schema，table信息

information\_schema.tables 系统表，可以查询到某数据库中的schema、

#### 2.2.9.2 namespace

pg\_catalog.pg\_namespace 系统表，显示命名空间。

### 2.2.10 显示view/function的定义

#### 2.2.10.1 显示view定义

\sv [+] VIEWNAME

#### 2.2.10.1 显示function定义

\sf [+] FUNCNAME

## 2.3 库函数--FUNCTION

Reference: <https://www.postgresql.org/docs/9.1/static/functions-math.html>

### 2.3.1 substring(str, start[, end])

### 2.3.2 to\_number(text,text)

Reference: <https://www.techonthenet.com/postgresql/functions/to_number.php>

例子：

select max(to\_number) from

(select to\_number((substr),'00000') from

(select substr(id,5) from place.cities where id like '%CTW\_%') as subtable) as number

### 2.3.3 lower(str)

对于比较的字符进行小写转换

### 2.3.4 replace

update place.detailcityinfo set country=replace(country, ‘\s+’,’’)

### 2.3.5 trim

仅去除低端首尾的空格

update place.detailcityinfo set country=trim(country);

### 2.3.6 length | char\_length | bit\_length

### 2.3.7 overlay

类似与replace方法，将某个字符的部分转换成其他内容。

overlay('Txxxxas' placing 'hom' from 2 for 4)

蓝色字体为关键字。

### 2.3.8 round(v1, v2)

v2, optional， to decide precision of target value.

### 2.3.9 cbrt (v1, v2) ----cube root

### 2.3.10 sqrt (v1, v2) ----square root

### 2.3.11 System Management Function

#### 2.3.11.1 pg\_column\_size----calculating size of column

unit: byte

e.g: select pg\_ column \_size(databasename\_string);

#### 2.3.11.2 pg\_database\_size----calculating size of database

unit: byte

e.g: select pg\_database\_size(databasename\_string);

#### 2.3.11.3 pg\_ls\_dir

select pg\_ls\_dir(‘./’); //表示postgres的安装路径(/usr/local/var/postgres/)

#### 2.3.11.4 pg\_terminate\_backend(pid int)

终止a backend.

#### 2.3.11.5 系统表

2.3.11.5.1 pg\_stat\_activity

select \* from pg\_stat\_activity; //显示了所有的进程活动状态

2.3.11.5.2 pg\_stat\_database

#### 2.3.11.6 pg\_backend\_pid

select pg\_backend\_pid(); //获取当前session的pid

## 2.4 Aggregate

Group by 用来进行数据分组，对于aggregate function是可以应用在group by之后的分组里的。Having 则可以对于分组进行过滤。

## 2.5 foreign key

保证referential integrity of data, 当外键存在时，此时插入数据时，要求外键表明的行必须在依赖的表中。

## 2.6 cluster

### 2.6.1 introduction

In file system terms, a database cluster is a single directory under which all data will be stored. We call this the data directory or data area. It is completely up to you where you choose to store your data. There is no default, although locations such as /usr/local/pgsql/data or /var/lib/pgsql/data are popular.

# 存储过程(程序)/函数

## json写入数据表中

参考：<https://www.postgresql.org/docs/9.3/static/functions-json.html>

储存过程举例：

DECLARE lat\_change NUMERIC(15, 10);

DECLARE lng\_change NUMERIC(15, 10);

DECLARE lat\_min NUMERIC(15,10);

DECLARE lng\_min NUMERIC(15,10);

DECLARE lat\_max NUMERIC(15,10);

DECLARE lng\_max NUMERIC(15,10);

BEGIN

--SELECT distance;

lat\_change = distance / 111.2;

lng\_change = abs(cos(lat\*(pi()/180)));

lat\_min = lat - lat\_change;

lat\_max = lat + lat\_change;

lng\_min = lng - lng\_change;

lng\_max = lng + lng\_change;

RETURN ST\_MakeBox2D(POINT(lat\_min,lng\_min)::geometry, POINT(lat\_max,lng\_max)::geometry);

--RETURN box((lat\_min, lng\_min), (lat\_max, lng\_max));

END;

## 3.1 postgres提供的系统函数

### 3.1.1 关于location 函数集合

参考：<https://postgis.net/docs/index.html>

#### 3.1.1.1 ST\_XMin

使用一个box2d、box3d或者geometry(几何图行)的最小集合

参考：

let fnStr = `city."getBoundsFromLatLng"(${point[1]}, ${point[0]}, ${distance})`;

let sql = `SELECT ST\_XMin(${fnStr}) as lat\_min, ST\_XMax(${fnStr}) as lat\_max, ST\_YMin(${fnStr}) as lng\_min, ST\_YMax(${fnStr}) as lng\_max`;

# 4. 事务处理

操作原子性

Window function：

语法：

在现有数据下，执行某个函数，将相应的结果添加显示在新列。如何计算每个部门的平均工资。

Select depname, empno, salary, avg(salary) over (partition by depname) from empsalary

每个window function其后总会跟随over子句，用于决定window function在执行操作时，各行如何被划分成组。

关键字partition by用于分组。但是order by 和 partition by某些情况下可以被忽略。

同一个query中可以执行多个 window function，但是其作用的数据都是同一个virtual table

Window function只存在select和order by从句中。

注意：

Window function相结合的是window frame(分区)，window function的执行作用域就是window frame， 默认是whole rows，即over为空的情况。

# 5. 部署问题

## 5.1安装

API安装： apt-get install postgresql-9.6

图形管理界面： apt-get install pgadmin3

命令行： su – postgres

## 5.2 开放数据库端口和链接ip

### 5.2.1 postgresql.conf

修改postgres服务器的监听地址: listen\_address ‘\*’ 或者 listen\_address ‘0.0.0.0/0’

### 5.2.2 pg\_hba.conf

客户端验证配置文件。原命令为host-based authentication(依赖主机的验证);

需要配置的项： type、database、user、address、method.

其中对于链接方式:

---md5: 表示链接输入的密码使用md5进行加密

---peer：仅适用于本地，本地系统的用户名与请求的数据库的user相同。

### 5.2.3 pg\_ctl.conf

配置数据库的集群初始化、数据库服务器的启动、停止、重启， 显示运行数据库的状态。

#### 5.2.3.1 stop mode

pg\_ctl stop

四种停止数据库方式： 使用-m 参数形式。

Smart: 等待所有链接断开，备份完成。当处于热排队阶段，复原、流化副本在所有客户端断开链接后，终止。

Fast：不会等待链接断开和网上备份完成。存在事务的，将会rollback，同时断开链接。

Immediate：快速终止进程。下次重启时造成恢复问题。

#### 5.2.3.2 reload

形同nginx， 通过reload方式重新加载配置文件进行数据库启动。

#### 5.2.3.3

### 5.2.4 pg\_ident.conf

配置linux系统用户和数据库用户的同名映射，完成peer方式的验证。

# 6. sql的速度

## 6.1 join

### 6.1.1 left join | right join | inner join

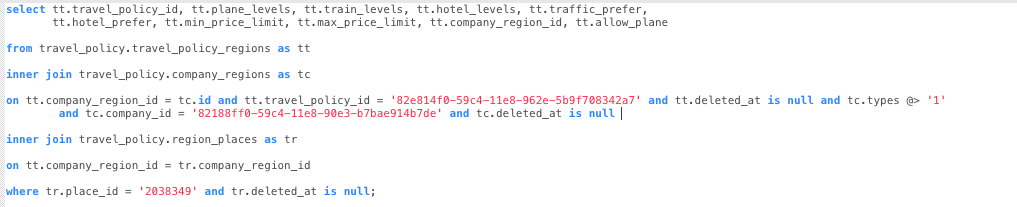
#### 6.1.1.1 left join

左连接，重点在左，对于左侧数据表，每一行都会在结果中体现，右侧表无matching的row，使用null替代

#### 6.1.1.2 right join

#### 6.1.1.3 inner join

结果集中左右两张表数据都存在，且相互关联



select tt.travel\_policy\_id, tt.plane\_levels, tt.train\_levels, tt.hotel\_levels, tt.traffic\_prefer, tt.hotel\_prefer, tt.min\_price\_limit, tt.max\_price\_limit, tt.company\_region\_id, tt.allow\_plane

from travel\_policy.travel\_policy\_regions as tt

inner join travel\_policy.company\_regions as tc on tt.company\_region\_id = tc.id and tt.travel\_policy\_id = '82e814f0-59c4-11e8-962e-5b9f708342a7' and tt.deleted\_at is null and tc.types @> '1' and tc.company\_id = '82188ff0-59c4-11e8-90e3-b7bae914b7de' and tc.deleted\_at is null

inner join travel\_policy.region\_places as tr on tt.company\_region\_id = tr.company\_region\_id where tr.place\_id = '1814991' and tr.deleted\_at is null;

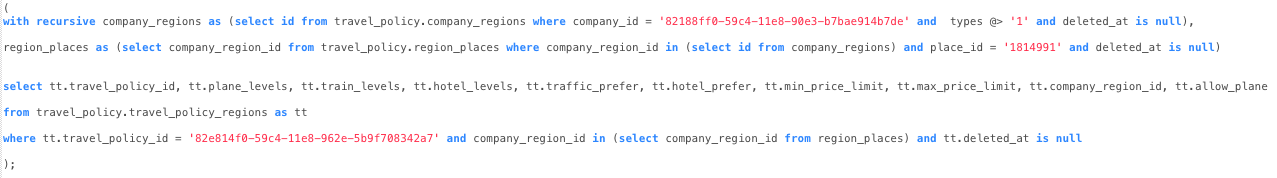
travel\_policy\_region表数据1591

company\_regions表数据 2211

region\_places 表数据7984

总耗时：0.07sec

##### 对比 with clause



(with company\_regions as (select id from travel\_policy.company\_regions where company\_id = '82188ff0-59c4-11e8-90e3-b7bae914b7de' and types @> '1' and deleted\_at is null),

region\_places as (select company\_region\_id from travel\_policy.region\_places where company\_region\_id in (select id from company\_regions) and place\_id = '1814991' and deleted\_at is null)

select tt.travel\_policy\_id, tt.plane\_levels, tt.train\_levels, tt.hotel\_levels, tt.traffic\_prefer, tt.hotel\_prefer, tt.min\_price\_limit, tt.max\_price\_limit, tt.company\_region\_id, tt.allow\_plane

from travel\_policy.travel\_policy\_regions as tt

where tt.travel\_policy\_id = '82e814f0-59c4-11e8-962e-5b9f708342a7' and company\_region\_id in (select company\_region\_id from region\_places) and tt.deleted\_at is null );

travel\_policy\_region表数据1591

company\_regions表数据 2211

region\_places 表数据7984

总耗时：0. 05 ~ 0.10 sec , 总耗时比inner join 慢一倍左右

高级语法