[1. IDE 1](#_Toc110)

[Navicat的使用 1](#_Toc27385)

[基本命令 1](#_Toc25186)

[2.1 基本 1](#_Toc23803)

[2.1.1 修改表结构 2](#_Toc30944)

[2.1.2 Join & union 2](#_Toc14229)

[2.1.3 关键字 2](#_Toc25594)

[2.1.4 数据结构 2](#_Toc1585)

[2.2 命令行操作 3](#_Toc18058)

[2.3 库函数 3](#_Toc30962)

[2.3.1 substring(str, start[, end]) 3](#_Toc3603)

[2.3.2 to\_number(text,text) 3](#_Toc30098)

[2.3.3 lower(str) 3](#_Toc1697)

[2.3.4 replace 4](#_Toc2110)

[2.3.5 trim 4](#_Toc692)

[2.3.6 length | char\_length | bit\_length 4](#_Toc31836)

[2.4 Aggregate 4](#_Toc24441)

[2.5 foreign key 4](#_Toc25683)

[3. 存储程序 4](#_Toc12257)

[json写入数据表中 4](#_Toc20692)

[4. 事务处理 4](#_Toc27988)

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;

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

# 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;

### 2.1.1 修改表结构

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

### 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

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。

**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。

**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';

**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

**->**：

**->>**：

**#>：类似嵌套分层级进行查询**

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。

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

## 2.2 命令行操作

### 2.2.1 命令行登陆

Psql -h ip -d database -U

**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)

## 2.3 库函数

### 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。

## 2.4 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。

# 存储程序

## json写入数据表中

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

# 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。