Qunar

本指南目的在于经验总结， 帮助PostgreSQL使用者快速入门，在开发及管理过程中有章可循.

1 命名规范

1. DB object: database, schema, table, column, view, index, sequence, function, trigger等名称
(1) 建议使用小写字母、数字、下划线的组合
(2) 建议不使用双引号即"包围，除非必须包含大写字母或空格等特殊字符
(3) 长度不能超过63个字符
(4) 不建议以pg开头（避免与系统DB object混淆），不建议以数字开头
(4) 禁止使用 SQL 关键字，例如 type, order 等

2. table能包含的column数目,根据字段类型的不同，数目在 250 到 1600 之间

3. 临时或备份的DB object:table,view 等,建议加上日期, 如dba_ops.b2c_product_summay_2014_07_12 (其中dba_ops 为DBA专用schema)

4. index命名规则为: 表名_列名_idx,如student_name_idx,建议不显式给出index name,使用DBMS默认给出的index name,如create index ON student (name),则默认给出的index名称为student_name_idx

2 Column设计

1. 建议能用数值类型的，就不用字符类型

2. 建议能用varchar(N) 就不用char(N),以利于节省存储空间

3. 建议能用varchar(N) 就不用text,varchar

4. 建议使用default NULL,而不用default '',以节省存储空间

5. 建议使用ip4,ip4r,ip6,ip6r,ipaddress,iprange 来存储IP,IP范围；使用macaddr来存储MAC (Media Access Control) address

6. 建议使用timestamp with time zone(timestamptz),而不用timestamp without time zone,避免时间函数在对于不同时区的时间点返回值不同,也为业务国际化扫清障碍

7. 建议使用NUMERIC(precision, scale)来存储货币金额和其它要求精确计算的数值, 而不建议使用real, double precision

8. 建议使用hstore 来存储非结构化,key-value 键值型,对数不定的数据

9. 建议使用ltree 来存储 Top.中国.北京.海淀区.苏州街.维亚大厦 这种 树状层次结构 数据

10. 建议使用jsonb(比json更有优势)来存储JSON (JavaScript Object Notation) data

11. 建议使用Geometric Types 结合PostGIS来实现地理信息数据存储及操作

12. 建议使用如下range类型代替字符串或多列来实现范围的存储

```
int4range — Range of integer

int8range — Range of bigint

numrange — Range of numeric

tsrange — Range of timestamp without time zone

tstzrange — Range of timestamp with time zone

daterange — Range of date
```

3 Constraints 设计

1. 建议每个table都有主键;

2. 建议不要用有业务含义的名称作为主键,比如身份证或者国家名称,尽管其是unique的

3. 建议主键的一步到位的写法:id serial primary key 或id bigserial primary key

4. 建议内容系统中size较大的table主键的等效写法如下,便于后续维护

```
create table test(id serial not null );

create unique index CONCURRENTLY ON test (id);
```

4 Index 设计

1. PostgreSQL 提供的index类型: B-tree, Hash, GiST (Generalized Search Tree), SP-GiST (space-partitioned GiST), GIN (Generalized Inverted Index), BRIN (Block Range Index),目前不建议使用Hash

2. 建议create 或 drop index 时,加 CONCURRENTLY参数,这是个好习惯，达到与写入数据并发的效果

3. 建议对于频繁update, delete的包含于index 定义中的column的table, 用create index CONCURRENTLY , drop index CONCURRENTLY 的方式进行维护其对应index

4. 建议用unique index 代替unique constraints,便于后续维护

5. 建议对where 中带多个字段and条件的高频 query，参考数据分布情况，建多个字段的联合index

6. 建议对固定条件的（一般有特定业务含义）且选择比好（数据占比低）的query，建带where的Partial Indexes

```
select * from test where status=1 and col=?; -- status=1
create index on test (col) where status=1;
```

7. 建议对经常使用表达式作为查询条件的query，可以使用表达式或函数索引加速query

```
select * from test where exp(xxx);
create index on test ( exp(xxx) );
```

8. 建议不要建过多index，一般不要超过6个，核心table（产品，订单）可适当增加index个数

5 关于NULL

1. NULL 的判断：IS NULL ，IS NOT NULL

2. 注意boolean 类型取值 true，false， NULL

3. 小心NOT IN 集合中带有NULL元素

```
mydb=# SELECT * FROM (VALUES(1),(2)) v(a) ;
 a
---
 1
 2
(2 rows)


mydb=# select 1 NOT IN (1, NULL);
 ?column?
----------
 f
(1 row)


mydb=# select 2 NOT IN (1, NULL);
 ?column?
----------

(1 row)

mydb=# SELECT * FROM (VALUES(1),(2)) v(a) WHERE a NOT IN (1, NULL);
 a
---
(0 rows)

--SELECTWHEREtrue
```

4. 建议对字符串型NULL值处理后，进行 || 操作

```
mydb=# select NULL||'PostgreSQL';
 ?column?
----------

(1 row)

mydb=# select coalesce(NULL,'')||'PostgreSQL';
  ?column?
------------
 PostgreSQL
(1 row)
```

5. 建议对hstore 类型进行处理后，进行 || 操作，避免被NULL吃掉

```
mydb=# select  NULL::hstore || ('key=>value') ;
 ?column?
----------

(1 row)

mydb=# select  coalesce(NULL::hstore, hstore(array[]::varchar[])) || ('key=>value') ;
    ?column?
----------------
 "key"=>"value"
(1 row)

mydb=# select  coalesce(NULL::hstore,''::hstore) || ('key=>value') ;
    ?column?
----------------
 "key"=>"value"
(1 row)
```

6. 建议使用count(1) 或count(*) 来统计行数，而不建议使用count(col) 来统计行数，因为NULL值不会计入

注意: count(多列列名)时，多列列名必须使用括号，例如count((col1,col2,col3));
注意多列的count，即使所有列都为NULL，该行也被计数，所以效果与count(*) 一致

7. count(distinct col) 计算某列的非NULL不重复数量，NULL不被计数

注意: count(distinct (col1,col2,...)) 计算多列的唯一值时，NULL会被计数，同时NULL与NULL会被认为是相同的

8. NULL 的count与sum

```
select count(1), count(a), sum(a)  from (SELECT * FROM (VALUES (NULL), (2) ) v(a)) as foo where a is
NULL;
 count | count | sum
-------+-------+-----
     1 |     0 |
(1 row)
```

9. 判断两个值是否相同（将NULL视为相同的值）

```
 select null is distinct from null;
 ?column?
----------
 f
(1 row)

 select null is distinct from 1;
 ?column?
----------
 t
(1 row)

select null is not  distinct from null;
 ?column?
----------
 t
(1 row)

mydb=#  select null is not distinct from 1;
 ?column?
----------
 f
(1 row)
```

6 开发相关规范

1. 建议对DB object 尤其是COLUMN 加COMMENT，便于后续新人了解业务及维护

2. 建议非必须时避免select *，只取所需字段，以减少包括不限于网络带宽消耗，避免表结构变更对程序的影响（比如某些prepare query）

3. 建议update 时尽量做 <> 判断,比如update table_a set column_b = c where column_b <> c

4. 建议将单个事务的多条SQL操作,分解、拆分，或者不放在一个事务里，让每个事务的粒度尽可能小，尽量lock少的资源，避免lock 、dead lock的产生

5. 建议 大批量的数据入库时， 使用copy ，不建议使用insert，以提高写入速度

6. 建议对报表类的或生成基础数据的查询，使用物化视图(MATERIALIZED VIEW)定期固化数据快照，
避免对多表（尤其多写频繁的表）重复跑相同的查询，且物化视图支持 REFRESH MATERIALIZED VIEW CONCURRENTLY, 支持并发更新

7. 建议复杂的统计查询可以尝试窗口函数 Window Functions

8. state 为 idle in transaction 的连接，如果出现在Master, 会无谓的lock住相应的资源, 可导致后续产生lock,甚至deadlock; 出现在Slave, 可导致卡住主从同步

```
a. pythonpsycopg2 postgresqlautocommitfalse, ,slave transaction, true

b. psql  pgadminDB, begin;  xxx; commit;
```

9. 建议在 MyBatis框架中使用SQL语句时，不要写 useGeneratedKeys="true"， 或直接设置 useGeneratedKeys="false"，避免returning *的产生，从而浪费带宽，如果需要返回某个字段，比如id，可以按照如下设置

```
StatementHandlerPreparedStatementHandle

useGeneratedKeys="true"; keyColumn="id"; keyProperty="id"
```

10. 建议发给PostgreSQL DBA 发布的DDL，附带常用DML: SELECT, INSERT ,DELETE, UPDATE，便于DBA给出create index CONCURRENTLY 等其他优化建议

7 管理相关规范

1. 建议清空表时，使用truncate，不建议使用delete

2. 建议向大size的table中add column时，将 alter table t add column col datatype not null default xxx；分解为如下，避免填充default值导致的过长时间锁表

```
alter table t add column col datatype

alter table t alter column col set default xxx

update t set column = default where id = 1;

......
......
......

update t set column = default where id = N;

------,\watch
------ update table t  set column= DEFAULT where id in ( select id from t where column is null limit
1000 ) ; \watch 3

alter table t alter column col set not null
```

3. 建议执行DDL,比如CRAETE、DROP、ALTER 等, 尤其多条， 不要显式的开transaction, 因为加lock的mode非常高,极易产生deadlock

4. 建议对频繁更新的表，建表时指定fillfactor，一般建议值为85

```
create table test(id int ) with(fillfactor=85);
```

5. 建议运行在SSD上的实例, random_page_cost (默认值为4) 设置为1.0~2.0之间, 使查询规划器更倾向于使用索引扫描

6. 建议在需要使用explain analyze 查看实际真正执行计划与时间时，如果是写入 query，强烈建议先开启事务， 然后回滚