**命名规范**

【强制】对象名要求可读性高(一般要求用英文)，让别人一看命名，就知道这个字段表示什么意思。

【强制】库名、表名限制命名长度，建议表名及字段名字符总长度小于等于63。

【强制】对象名（表名、列名、函数名、视图名、序列名、等对象名称）规范，对象名务必只使用小写字母，下划线，数字。不要以pg开头，不要以数字开头，不要使用保留字。不要使用双引号包围，除非必须包含大写字母或空格等特殊字符。

保留字参考： http://www.postgres.cn/docs/14/sql-keywords-appendix.html

【强制】index命名规则为：表名\_列名\_idx，如student\_name\_idx，建议不显示给出index\_name，使用PostgreSQL默认给出的index\_name，create index on student(name); 生成的索引名为student\_name\_idx。

【强制】query中的别名不要使用 "小写字母，下划线，数字" 以外的字符，例如中文。

【推荐】主键索引应以 pk\_ 开头， 唯一索引要以 uk\_ 开头，普通索引要以 idx\_ 打头。

【推荐】临时表以 tmp\_ 开头，子表以规则结尾，例如按年分区的主表如果为t1, 则子表为t1\_2016，t1\_2017。

【推荐】库名最好与应用名称一致，便于辨识。

【推荐】不建议使用public schema(不同业务共享的对象可以使用public schema)，应该为每个应用分配对应的schema，schema\_name最好与username一致。

【推荐】comment不要使用中文，因为编码可能不一样，如果存进去和读取时的编码不一致，导致可读性不强。 pg\_dump时也必须与comment时的编码一致，否则可能乱码。

以下对象命名采用固定前缀进行命名，前缀表示数据库对象的类型，前缀代码规范如下：

|  |  |  |
| --- | --- | --- |
| 类型 | 前缀规范 | 说明 |
| 表空间 | ts\_ | Tablesapce |
| 数据库 | db\_ | xxdb | database |
| 表 | t\_ | table |
| 列 | 无 | 无前缀 |
| 临时表 | tmp\_ | temporary table |
| 分区 | pt\_ | partition |
| 视图 | vw\_ | view,v\_用于表示变量 |
| 物化视图 | mv\_ | material view |
| 索引 | idx\_ | Index缩写，不区分索引类型。约束型索引参照约束命名 |
| 主键约束 | pk\_ | primary key |
| 外键约束 | fk\_ | foreign key |
| 唯一约束 | uk\_ | unique key |
| 序列 | seq\_ | sequence |
| 函数 | f\_ | function |
| 存储过程 | sp\_ | store procedure |
| 触发器 | trg\_ | trigger |
| 普通变量 | v\_ | variable |
| 游标变量 | cur\_ | cursor |
| 输入参数 | p\_ | parmameter |
| 输出参数 | o\_ | out |

**Column设计规范**

【强制】表结构中字段的数据类型与应用程序中的定义保持一致，字段排序规则也需要保持一致，避免报错或无法使用索引的情况发生。

【推荐】能用数值类型的，就不用字符类型，使用好的数据类型，可以提高数据的查询效率。

【推荐】能用varchar(N)的，就不用char(N)，以利于节省存储空间

【推荐】能用varchar(N)的，就不用text、varchar

【推荐】能用 default NULL，就不用 default ''，以节省存储空间

【推荐】使用 cidr、 inet 来存储IP；使用 macaddr 来存储MAC address

【推荐】使用 timestamp with time zone ，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

【推荐】使用 numeric(precision,scale) 来存储货币金额和其他要求精确计算的数值，而不建议 real、 double precision

【推荐】使用JSONB(比JSON更有优势)来存储JSON data

【推荐】使用 Geometric Types 结合 PostGIS 来实现地理信息数据存储及操作

【推荐】使用 range 类型代替字符串或多列来实现范围的存储

int4range — integer的范围

int8range —bigint的范围

numrange —numeric的范围

tsrange —timestamp without time zone的范围

tstzrange —timestamp with time zone的范围

daterange —date的范围

【推荐】用户在设计表结构时，建议规划好，避免经常需要添加字段，或者修改字段类型或长度。

**Constraints设计规范**

【强制】使用外键时，如果你使用的PG版本没有自动建立fk的索引，则必须要对foreign key手工建立索引，否则可能影响references列的更新或删除性能。

【强制】使用外键时，一定要设置fk的action，例如cascade，set null，set default。

create table t2(id int references t1(id) on delete cascade on update cascade, info text);

【推荐】每个table都有主键

【推荐】不要用有业务含义的名称作为主键，比如身份证或国家名称，尽管其是 unique 的。

【推荐】主键的写法一步到位：id serial primary key 或 id bigserial primary key

**Index设计规范**

【强制】btree索引字段不建议超过2000字节，如果有超过2000字节的字段需要建索引，建议使用函数索引 或 哈希索引，或者使用分词索引。

【推荐】PostgreSQL提供的index类型：B-tree、Hash、GIST、SP-GIST、GIN、BRIN，目前不建议使用Hash

【推荐】对于值与堆表的存储顺序线性相关的数据，如果通常的查询为范围查询，建议使用BRIN索引。

例如流式数据，时间字段或自增字段，可以使用BRIN索引，减少索引的大小，加快数据插入速度。

create index idx on tbl using brin(id);

【推荐】create 或 drop index 时，加上 CONCURRENTLY 参数，这是个好习惯，达到写入数据并发的效果

【推荐】用create unique index 代替 unique constaint，便于后续维护

【推荐】对where中带多个字段and条件的高频query，根据数据分布情况，建多列索引

【推荐】对固定条件的(一般有特点业务含义)且选择比好(数据占比低)的query，建带where的 Partial index

select \* from test where status=1 and col=?

create index on test(col) where status=1;

【推荐】对于经常使用表达式作为查询条件的语句，可以使用表达式或函数索引加速查询。

select \* from t1 where exp(xxx);

create index idx on t1( exp );

【推荐】不用建过多index，一般不要超过6个，因为创建过多的索引，会降低写得速度。核心table(产品、订单)可适当增加index个数。

【推荐】应该尽量避免全表扫描(除了大数据量扫描的数据分析)，PostgreSQL支持几乎所有数据类型的索引。

【推荐】当业务有中文分词的查询需求时，建议使用PostgreSQL的分词插件zhparser。建议在分词字段使用gin索引，提升分词匹配的性能。

【推荐】当用户有正则表达式查询，或者文本模糊查询的需求时，建议对字段使用trgm的gin索引，提升模糊查询或正则表达式匹配的查询效率，同时覆盖了前后模糊的查询需求。如果没有创建trgm gin索引，则不推荐使用前后模糊查询例如like %xxxx%。

【推荐】当用户有prefix或者 suffix的模糊查询需求时，可以使用索引，或反转索引达到提速的需求。

select \* from t1 where col ~ '^abc'; --前缀查询

select \* from t2 where reverse(col) ~ '^fed'; – 后缀查询使用反转函数索引

【推荐】评估表数据量。如果表数据量只有一百几十行，就没有必要加索引。

**关于NULL的注意事项**

1.NULL的判断：is null、is not null

2.注意boolean类型取值 ture、false、null

3.小心not in 集合中带有null元素

4.建议对字符串型null值处理后，进行||操作

select NULL||'PostgreSQL';--NULL

5. 建议使用count(1) 或 count(\*)来统计，而不建议使用count(col)来统计，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。

6.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

7.null的count和sum

8.null = null ?

**开发相关规范**

【强制】对于频繁更新的表，建议建表时指定表的fillfactor=85，每页预留15%的空间给HOT更新使用。

create table t1(id int, info text) with(fillfactor=85);

【推荐】对数据库对象，尤其是 column 加 comment，便于后续新人了解业务及维护

【推荐】非必要时避免使用 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

【推荐】update时尽量做 <> 判断，比如update table t1 set col1 = xx where id=xx and col1 <> xx;

【推荐】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

【推荐】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

【推荐】对报表类的或生成基础数据的查询，使用物化视图定期固化快照，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。

【推荐】对复杂的统计查询可以尝试窗口函数

【推荐】state 为 idle in transaction 的连接，如果出现在master，会lock住相应的资源，可导致后续产生lock，甚至dead lock；出现在slave，可导致卡住主从同步

【推荐】保证分区表的主键序列全局唯一。使用多个序列，每个序列的步调不一样，或者每个序列的范围不一样即可。

create sequence seq\_t1 increment by 3 start with 1;

create sequence seq\_t2 increment by 3 start with 2;

create sequence seq\_t3 increment by 3 start with 3;

create table t1 (id int primary key default nextval('seq\_t1') check(mod(id,3)=1), info text);

create table t2 (id int primary key default nextval('seq\_t2') check(mod(id,3)=2), info text);

create table t3 (id int primary key default nextval('seq\_t3') check(mod(id,3)=0), info text);

create sequence seq\_t1 increment by 1 minvalue 1 maxvalue 100000000 start with 1 no cycle ;

create sequence seq\_t2 increment by 1 minvalue 100000001 maxvalue 200000000 start with 100000001 no cycle ;

create sequence seq\_t3 increment by 1 minvalue 200000001 maxvalue 300000000 start with 200000001 no cycle ;

create table t1(id int primary key default nextval('seq\_t1') check(id >=1 and id<=100000000), info text);

create table t2(id int primary key default nextval('seq\_t2') check(id >=100000001 and id<=200000000), info text);

create table t3(id int primary key default nextval('seq\_t3') check(id >=200000001 and id<=300000000), info text);

【推荐】为了全球化的需求，所有的字符存储与表示，均以UTF-8编码，占用1-4个字节。

select length('阿里巴巴');

select octet\_length('阿里巴巴');

字符串排序规则LC\_COLLATE推荐使用 'C'，而非 UTF8。LC\_COLLATE=UTF8 性能相对差一些，并且索引需要明确指定UTF8 pattern ops，才能支持LIKE查询。

【推荐】对于网络复杂并且RT要求很高的场景，如果业务逻辑冗长，应该尽量减少数据库和程序之间的交互次数，尽量使用数据库存储过程(如plpgsql)，或内置的函数。

【推荐】应用应该尽量避免使用数据库触发器，这会使得数据处理逻辑复杂，不便于调试。

【推荐】用户应该对频繁访问的大表进行分区，从而提升查询的效率、更新的效率、备份与恢复的效率、建索引的效率等等。

【推荐】不需要严格遵守 3NF，通过业务字段冗余来减少表关联

什么是数据库三范式（3NF）？

第一范式：对属性的原子性，要求属性具有原子性，不可再分解；

第二范式：对记录的唯一性，要求记录有唯一标识，即实体的唯一性，即不存在部分依赖；

第三方式：对字段的冗余性，要求任何字段不能由其他字段派生出来，它要求字段没有冗余，即不存在传递依赖

设计表及其字段之间的关系, 应尽量满足第三范式。但是有时候，可以适当冗余，来提高效率。

比如以下这张表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 商品名称 | 商品型号 | 单价 | 数量 | 总金额 |
| 手机 | 华为 | 8000 | 5 | 40000 |

以上这张存放商品信息的基本表。总金额这个字段的存在，表明该表的设计不满足第三范式，因为总金额可以由单价\*数量得到，说明总金额是冗余字段。但是，增加总金额这个冗余字段，可以提高查询统计的速度，这就是以空间换时间的作法。

这只是个小例子，开发设计的时候，要结合具体业务分析。

**管理相关规范**

【推荐】清空表时，使用truncate，不使用delete，减少wal log的产生

【推荐】PostgreSQL 11版本之前，向大size的table中 add column 时，将 alter table t1 add column col datatype not null default xxx; 分解为如下，避免填充default值导致的过长时间的锁表

alter table t1 add column col datatype；

alter table t1 alter column col set default xxx;

update t1 set col=default where …;--批量更新添加的列

alter table t1 alter column set not null;

【推荐】执行DDL，比如CREATE、DROP、ALTER等，尤其多条，不要显示的开transaction，因为lock的mode非常高，极易产生deadlock

【推荐】运行在SSD上的示例 random\_page\_cost(默认值为4)设置为1.1，使查询规划器更倾向于使用索引扫描

【推荐】在需要使用 explain analyze 查看实际真正执行与时间时，如果是写入query，强烈建议先开事务，然后回滚。

【推荐】如果应用经常要访问较大结果集的数据（例如100条），可能造成大量的离散扫描，建议想办法将数据进行聚集。

【推荐】未使用的大对象，一定要同时删除数据部分，否则大对象数据会一直存在数据库中，与内存泄露类似。

vacuumlo可以用来清理未被引用的大对象数据。