**全文检索介绍**

文本搜索操作符已经在数据库中存在很多年了，PostgreSQL对文本数据类型提供了~、~\*、LIKE和ILIKE操作符。

**文本搜索缺点：**

1.没有语义支持，即使是英语。

比如，satisfies和satisfy，当使用正则表达式寻找satisfy时，并不会查询到包含satisfies的文档。

2.没有对搜索结果的分类（排序）。当搜索出成千的文档时，查找效率很低。

3.由于没有索引的支持，每一次的搜索需要遍历所有的文档，整体搜索比较缓慢。

使用全文检索可以对文档进行预处理，使后续的搜索更快速。它的任务是从文档集中选择与搜索查询匹配的那些文档。（如果有很多匹配的文档，那么找到最佳的匹配很重要。）

**预处理过程包括：**

1.将文档解析成token。

实际上就是对文档分词。

2.将token转换为词素。

词素是标准化处理过的字符串。标准化通常包括：将大写字母折成小写字母、删除后缀（如英语中的s或者es）、删除停用词。 PostgreSQL使用词典执行这一步。词典可以定义停用词。

例如，单词通常会转换为小写字母，并且变量的结尾会被截断：

select to\_tsvector('There was a crooked man, and he walked a crooked mile');

还可以看到，某些单词（称为停用词）被完全丢弃（there, was, a, and, he），因为它们可能出现得太频繁，搜索它们没有意义。

3.保存优化后的预处理文档。

每个文档为标准化词素的有序组合，文档包含的查询词越密集其排序越高。

**全文检索数据类型**

PostgreSQL提供了两种数据类型用于支持全文检索。

tsvector类型用于存储预处理文档，其中包含词素及其在文档中的位置。tsvector把一个字符串按照空格进行分词，在进行分词处理的时候tsvector会自动去掉分词中重复的词条，按照一定的顺序录入。排序和去重是在输入期间自动完成的。

select 'There was a crooked man, and he walked a crooked mile'::tsvector;

对于英文全文检索，上面的单词会被认为非规范化的，需要通过to\_tsvector函数对这些单词进行规范化处理：

select to\_tsvector('There was a crooked man, and he walked a crooked mile');

select to\_tsvector('english','There was a crooked man, and he walked a crooked mile');--指定分词器

第一个参数指定分词器，第二个参数是要被分词的字符串。

查看所有分词器

\dF

查看默认分词器

show default\_text\_search\_config;

 default\_text\_search\_config

----------------------------

 pg\_catalog.english

tsquery用于存储查询条件。大致上，一个查询由一个或几个连接的词素组成：and &，or |，not ！..我们也可以使用括号来阐明操作优先级。

select 'walking & running'::tsquery;

select 'walking | running'::tsquery;

select '! running'::tsquery;

select 'man & (walking | running)'::tsquery;

select 'man & walking & !running'::tsquery;

to\_tsquery函数会将单词转换为tsquery类型前进行规范化处理。

select to\_tsquery('walking & running');

前缀查询

select to\_tsquery('english', 'postgres:\*' );

 to\_tsquery

------------

 'postgr':\*

表示postgr开头的词素。

文本检索操作符和函数：http://www.postgres.cn/docs/14/functions-textsearch.html

其中最重要的是匹配运算符为@@。

select 'There was a crooked man, and he walked a crooked mile'::tsvector @@ 'he & (walking | running)'::tsquery;--false

select 'There was a crooked man, and he walked a crooked mile'::tsvector @@ 'he & (walked | ran)'::tsquery;--true

**文本匹配测试**

SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector @@ 'cat & rat'::tsquery;--true

SELECT 'fat & cow'::tsquery @@ 'a fat cat sat on amat and ate a fat rat'::tsvector ;--false

tsquery和tsvector未将文本中的词标准化，通常使用to\_tsvector和to\_tsquery解析和标准化文档字符串。

SELECT to\_tsvector('fat cats ate fat rats') @@to\_tsquery('fat & rat') ;--true

注意：下面这种方式是不可行的，由于tsvector没有对rats进行标准化，所以rats不匹配rat。

SELECT 'fat cats ate fat rats'::tsvector @@to\_tsquery('fat & rat');--false

**全文检索应用案例**

不使用索引进行全文检索：将info字段中包含science的每一行打印出来。

CREATE TABLE t1(id int, info text);

INSERT INTO t1 VALUES(1, 'Philology is the study of words, especially the history and development of the words in a particular language or group of languages.');

INSERT INTO t1 VALUES(2, 'Mathematics is the sciencethat deals with the logic of shape, quantity and arrangement.');

INSERT INTO t1 VALUES(3, 'Computer science is the study of processes that interact with data and that can be represented as data in the form of programs.');

INSERT INTO t1 VALUES(4, 'Chemistry is the scientificd is cipline involved with elements and compounds composed of atoms, molecules and ions.');

INSERT INTO t1 VALUES(5, 'Geography is a field of science devoted to the study of the lands, features, inhabitants, and phenomena of the Earth and planets.');

INSERT INTO t1 VALUES(6, 'History is a subject studied in schools, colleges, and universities that deals with events that have happened in the past.');

INSERT INTO t1 VALUES(7, 'Medical science is the science of dealing with the maintenance of health and the prevention and treatment of disease.');

INSERT INTO t1 VALUES(8, 'Physics is one of the most fundamental scientific disciplines, and its main goal is to understand how theuniverse behaves.');

SELECT \* FROM t1 WHERE to\_tsvector('english', info) @@ to\_tsquery('english', 'science');

上面的查询指定english配置来解析和规范化字符串。当然也可以省略此配置，通过default\_text\_search\_config进行配置设置：

SHOW default\_text\_search\_config;

 default\_text\_search\_config

----------------------------

 pg\_catalog.english

SELECT \* FROM t1 WHERE to\_tsvector(info)@@ to\_tsquery('science');

检索出在info字段中包含treatment和science的记录：

SELECT \* FROM t1 WHERE to\_tsvector(info) @@ to\_tsquery('treatment & science');

上面均在没有索引，查询很慢。文本搜索在实际使用中通常需要创建索引。

**创建GIN索引**

CREATE INDEX ON t1 USING gin(to\_tsvector('english', info));

系统默认采用default\_text\_search\_config所指定的分词器。

创建索引时必须使用to\_tsvector指定分词器，否则索引内容可能不一致。只有指定了分词器名称的全文检索函数才可以在索引表达式中使用。因为索引的内容不受default\_text\_search\_config的影响。

查询时也要指定分词器数，且参数值与索引中相同时，才会使用该索引。例如，WHERE to\_tsvector('english', info) @@'treatment & science' 可以使用索引，但WHERE to\_tsvector(info) @@ 'treatment & science'不能使用索引。这确保索引各条目是使用相同的分词器创建的。

SELECT \* FROM t1 WHERE to\_tsvector(info) @@ to\_tsquery('treatment & science') ;

查看执行计划

查看执行计划，走了GIN索引，执行速度大幅提升

explain SELECT \* FROM t1 WHERE to\_tsvector(info) @@ to\_tsquery('treatment & science') ;

**另一个方法**

创建一个单独的tsvector列控制to\_tsvector的输出。

ALTER TABLE t1 ADD COLUMN info\_tsv tsvector;

UPDATE t1 SET info\_tsv = to\_tsvector('english',info);

或

ALTER TABLE t1 ADD COLUMN info\_tsv tsvector GENERATED ALWAYS AS(to\_tsvector('english', info)) STORED;

info\_tsv是一个 生成的列，它会自动与源数据同步。

然后为加速搜索创建一个GIN索引：

CREATE INDEX ON t1 USING gin(info\_tsv);

SELECT \* FROM t1 WHERE info\_tsv @@ to\_tsquery('science &Computer') ;

单独列的优势：

1.它不需要在查询时指定分词器，查询依赖于default\_text\_search\_config。

2.搜索比较快速，因为它不必重新利用to\_tsvector调用来验证索引匹配。

**中文全文检索**

PostgreSQL自带的parser插件适用于比较简单的语言（如英语），按照标点、空格切分语句即可获得有含义的词语，而中文比较复杂，词语之间没有空格分隔，长度也不固定，分词还和语义有关，因此parser不能用来做中文分词，建议使用zhparser。

zhparser是PostgreSQL的中文分词插件，安装后可以使PostgreSQL支持中文的全文检索（Full Text Search）。

下载地址：https://github.com/amutu/zhparser

国内站：https://gitee.com/mirrors/zhparser

安装zhparser之前需要先安装scws， 安装scws的方法如下：

wget http://www.xunsearch.com/scws/down/scws-1.2.3.tar.bz2

tar xvf scws-1.2.3.tar.bz2

cd scws-1.2.3

./configure

make

make install

安装zhparser的方法如下：

export PATH=/usr/local/pgsql/bin:$PATH

git clone https://gitee.com/mirrors/zhparser.git

cd zhparser

make

make install

使用zhparser的方法如下：

CREATE EXTENSION zhparser;

CREATE TEXT SEARCH CONFIGURATION testzhcfg (PARSER = zhparser);

ALTER TEXT SEARCH CONFIGURATION testzhcfg ADD MAPPING FOR n,v,a,i,e,l WITH simple;

注意， 这里创建了一个配置“testzhcfg”， 后续该配置会作为参数使用到函数“to\_tsvector”和“to\_tsquery”中。“n”“v”“a”“i”“e”“l”这几个字母分别表示一种Token策略， 只启用了这几种Token Mapping， 其他均被屏蔽。 具体支持的参数和含义可以用“\dFp+ zhparser”命令显示。

示例：

create table t1(id int,info text,info\_tsv tsvector GENERATED ALWAYS AS(to\_tsvector('testzhcfg', info)) STORED);

insert into t1(id,info) values('1', '国际足联世界杯（FIFA World Cup）');

insert into t1(id,info) values('2', '世界杯是由全世界国家级别球队参与，象征足球界最高荣誉，并具有最大知名度和影响力的足球赛事。');

insert into t1(id,info) values('3', '世界杯全球电视转播观众超过35亿 。世界杯每四年举办一次，任何国际足联会员国（地区）都可以派出代表队报名参加这项赛事。');

insert into t1(id,info) values('4', '巴西国家队是夺得世界杯冠军最多的球队（5次），并且在3夺世界杯后永久地保留了前任世界杯冠军奖杯雷米特杯。');

insert into t1(id,info) values('5', '现代足球发源地为英格兰 ，其代表队在1966年首夺世界杯。而现在的世界杯奖杯是大力神杯，由4夺世界杯冠军的德国在1974年首次捧杯并一直沿用。');

insert into t1(id,info) values('6', '2002年韩日世界杯，中国国家队首次晋级世界杯决赛圈 。');

insert into t1(id,info) values('7', '2022年卡塔尔世界杯于北京时间2022年11月21日至12月18日举行。');

insert into t1(id,info) values('8', '美国当地时间2022年6月16日，国际足球联合会（FIFA）宣布，将在美国、加拿大和墨西哥合办2026年世界杯。');

CREATE INDEX ON t1 USING GIN(info\_tsv);

使用中文的全文检索：

select \* from t1 where info\_tsv @@to\_tsquery('testzhcfg', '世界杯');

查询英文单词：

select \* from myblog wherecontent\_tsv @@ to\_tsquery ('testzhcfg', 'fifa');

检索以“世界”开头的文章：

select \* from t1 where info\_tsv @@ to\_tsquery ('testzhcfg', '世界');--没有记录

测试发现匹配不出来， 分词一般需要精确匹配， 如果分词中没有， 则无法查出，看看“国际足联世界杯”分词后是什么：

select to\_tsvector('testzhcfg', '国际足联世界杯');

发现分词中只有“世界杯”而没有“世界”所以无法查出， 但tsquery有一种前缀查询的语法：

select \* from myblog where info\_tsv @@ to\_tsquery('testzhcfg', '世界:\*');