**唯一索引**

1.创建唯一索引

CREATE UNIQUE INDEX ON t1(id);--单列

CREATE UNIQUE INDEX ON t1(id,name);--多列

用于强迫字段数值的唯一性，或者是多个字段组合值的唯一性，只有Btree支持唯一索引。

2.通过约束生成

唯一约束通过唯一索引实现。创建主键、唯一键时会隐含的创建唯一索引。

create table t1(id int primary key,name text,age int) ;

create table t1(id int unique,name text,age int) ;

外键约束依赖于被引用字段上的唯一约束（隐含的需要唯一索引）。

create table class(id int,name text);

create table student(id int,name text,age int,class\_id reference class(id));--ERROR

create unique index on class(id);

create table student(id int,name text,age int,class\_id reference class(id));--success

**表达式索引**

PostgreSQL支持表达式索引。

定义索引表达式后，当定义索引的表达式出现在SQL语句的WHERE子句或ORDERBY子句中时，PostgreSQL将考虑使用该索引。

注意：维护表达式的索引非常昂贵，因为每次在插入或更新的时候，都会去评估每一行的表达式并将其用于索引。如果当检索速度比插入和更新速度更重要时，应在表达式上使用索引。

在做大小写无关比较时， 常用的方法是使用lower函数：

SELECT \* FROM t1 WHERE lower(info) = 'hello world';

但因为使用了函数， 无法利用到 info 字段上的普通索引， 这时就需要建一个函数索引：

CREATE INDEX ON t1 (lower(info));

表达式上的索引并不是在索引查找时进行表达式的计算，而是在插入或更新数据行时进行计算， 因此在插入或更新时， 表达式上的索引会慢一些。

如果把表达式上的索引声明为UNIQUE，那么它会禁止向info列中插入只是大小写有区别而内容完全相同的数据行。因此，表达式上的索引可以实现简单唯一约束无法实现的一些约束。

CREATE UNIQUE INDEX ON mytest (lower(info));

**部分索引**

部分索引是对表中的部分数据进行索引， 是由一个条件表达式把这部分行筛选出来， 该条件表达式被称为部分索引的谓词。部分索引在一些情况下非常有用，有助于加快查询速度，同时减小索引的大小。

有时需要仅索引数据的一部分。这通常与高度不均匀的分布有关：通过索引搜索不频繁的值是有意义的，但通过对表的完全扫描更容易找到频繁值。

**定义部分索引的语法：**

CREATE INDEX index\_name ON table\_name(column\_list) WHERE condition;

WHERE子句指定应将哪些行添加到索引。

**示例一**： 设置部分索引以排除普通数值

create table users(id int,name text,age int);

建的部分索引：

create index on user(age) where age>=100;

如下SQL命令就可以使用到上面创建的部分索引

select \* from user where age>=100;

select \* from user where age=120;

而如下SQL命令就不能走到此部分索引，因为它的数据不在索引中：

select \* from user where age=88;

**示例二**： 设置部分索引以排除不感兴趣的数值

假如有一个表， 其中包含已付款和未付款的定单， 而未付款的定单占总表的一小部分， 并且是经常使用的部分，那么可以只在未付款定单上创建一个索引来改善查询性能。

表定义如下：

CREATE TABLE orders(order\_id int,amount decimal(12,2),status boolean);

创建的索引：

CREATE INDEX ON orders (order\_id) WHERE status is not true;

插入测试数据：

insert into orders select n, n\*1.1, true from generate\_series(1,1000000) n;

insert into orders select n, n\*1.1, false from generate\_series(1000001,1000020) n;

下面这条SQL语句就可以用到这个索引：

SELECT \* FROM orders WHERE status is not true AND order\_id > 5000000;

看一下执行计划， 命令如下：

explain SELECT \* FROM orders WHERE status is not true AND order\_id > 5000000;

该索引也可以用于那些完全不涉及索引键order\_id的查询， 示例如下：

explain SELECT \* FROM orders WHERE status is not true AND amount > 990000.00;

另一个方法：

在未付款定单上创建一个部分索引

CREATE INDEX ON orders (status) WHERE status=0;

只要WHERE子句出现 status=0，PostgreSQL就会考虑部分索引：

EXPLAIN SELECT \* FROM customer WHERE status = 0;

**示例三**： 设置一个部分唯一索引

部分索引可以在表的子集里创建唯一索引， 这样就满足部分数据的唯一性， 而并不约束那些不需要唯一的行。

假设有一个记录测试项目是否成功的表， 希望确保在每个目标和课题的组合中只有一条成功的记录， 但是可以有任意数量的不成功记录。 实现方法如下：

CREATE TABLE tests(subject text,target text,success boolean);

CREATE UNIQUE INDEX ON tests (subject,target) where boolean

如果只有少数测试的结果为成功， 而更多测试的结果为不成功，那么这将是一种非常高效的实现方法。

**多列索引**

多列索引是定义在表中多个字段上的索引，最多可包含32个字段。多列索引又称为复合索引、组合索引或串联索引。B-tree, GiST 和GIN索引支持多字段索引。适合多个字段条件的”AND”组合。

合适的多列索引效率比位图索引更高，因为很容易获得ctid，而不需要构建位图。

explain analyze select \* from users where name='zhangsan' and age=22;--位图索引

使用多列索引时列的顺序非常重要：页面内的数据将按第一个字段排序，然后按第二个字段排序，以此类推。

定义多列索引时，应将WHERE子句中经常使用的列放在列列表的开头，然后将在条件中不经常使用的列放在后面。

创建多列索引语法：

CREATE INDEX index\_name ON table\_name(a,b,c,...);

PostgreSQL优化器将在以下情况下考虑使用索引：

WHERE a = v1 and b = v2 and c = v3;

或

WHERE a = v1 and b = v2;

又或

WHERE a = v1;

但是，在以下情况下，它将不考虑使用索引：

WHERE  c = v3;

或者

WHERE b = v2 and c = v3;

**覆盖索引**

查询语句返回多列记录时，如果索引包含全部的返回列则不用回表获取数据，例如：

select name,age,score where name='zhangsan';

如果索引包含name,age,score三列则会用到 index only scan。

除此之外postgresql中还有一种方法可以实现类似的效果，即include子句。

index include即在索引中，放入额外属性内容，搜索时不需要回表，例如：

create index on student(name) include(age,score);--覆盖索引

create index on student(name,age,score);--多列索引

上面两张索引有什么区别呢？

索引1，KEY是name，在叶子节点中，存入KEY与(age,score)的内容。

索引2，KEY是(name,age,score)，在所有节点中，存储的都是所有字段的值，比索引1要重，包括空间，索引维护，更新等。

思考：唯一覆盖索引，还有唯一性约束吗

CREATE UNIQUE INDEX on t1(id) INCLUDE(name,age);  --id不能重复

**排他索引**

排他约束保证了两个行在指定的字段或表达式集合上使用指定操作符进行比较时，不重叠|相交。排他约束通过索引实现，在表中创建排他约束会隐含的创建索引。

**示例：会议室预约**

create extension btree\_gist;

create table room(id int,during tsrange,username text,EXCLUDE USING gist(roomid WITH =,during WITH &&));

张三先预订了9点到11点的101会议室

insert into room(101,'[2022-12-01 09:00, 2022-12-01 11:00)','zhangsan');

李四预定10点到12点的101会议室，由于和张三的预订时间重叠，预订失败！

insert into room values(101,'[2022-12-01 10:00, 2022-12-01 12:00)','lisi');--ERROR

李四更换到其它会议室，预订成功！

insert into room values(102,'[2022-12-01 10:00, 2022-12-01 12:00)','lisi');

对于没有排他约束功能的数据库，要完成同样的功能会比较麻烦！

注意：当所有操作符都是相等比较时，排他约束就相当于唯一约束，但普通的唯一约束会更快。

**示例：几何图形不相交**

create table circles(c circle,exclude using gist(c with &&));

insert into circles values('(1,3),4');

insert into circle values('(4,2),1');