PostgreSQL中json实现中的最大亮点：索引的“模式自由”

osdba · 2018-12-12 17:02:26发表 · 4826 次点击

1. 索引的模式自由介绍

我们知道在文档型数据库中最大的亮点是使用json实现了模式自由。模式自由也称为无模式，模式自由的最大好处就是不需要事先确定表结构就可以灵活的存储各种结构的数据，对开发人员来说就是“只要一提交代码就它可以很好的工作”，而原先的关系型数据库，需要事先设计表结构，对于一些特别灵活的业务就不是很方便。所以在一些关系型数据库中，逐渐的增加了对json的支持，如Oracle和MySQL。

但实际上不管是Oracle、MySQL，还是文档数据库如MongoDB，都不能做到索引的“模式自由”，即这些数据库中对json建索引都需要指定JSON中固定字段来建索引，查询时，只能按json中某个确定的字段条件来查询，MongoDB中虽然在json数据之上实现了全文检索，看起来象是一种“模式自由”，但是与按某个字段来查询还是有一些不同。

所以，在通常的数据库中，不管是关系型数据库Oracle、MySQL，还是文档型数据库如MongoDB，只是做到了表的模式自由，而没有做到索引的模式自由。而在PostgreSQL中通过GIN索引则实现了索引的模式自由，即索引时不需要指定JSON数据中的指定字段，后续就可以按JSON中已的任意的字段进行查询。

下面我们就详细介绍在PostgreSQL中如何用GIN索引实现索引的模式自由。

2. 创建测试数据

为了演示这个功能，我们需要建测试表。我们先建一个含有jsonb字段的表：

```
1. CREATE TABLE tbl_user_jsonb(id serial, user_info jsonb);
```

为了方便造数据，我们先造一个普通表，把数据生成到这张普通表中，然后利用PostgreSQL数据库提供的row_to_json的函数把普通表中的数据转换到json表中，这个功能也是PostgreSQL数据库的一个亮点。

```
1. CREATE TABLE tbl_user(
2.     id int4,
3.     user_id int8,
4.     user_name varchar(64),
5.     create_time timestamp(6) with time zone default clock_timestamp()
6. );
```

我们生成100万的数据，在上面的普通表中：

```
1. insert into tbl_user(id,user_id, user_name) select r,round(random()*2000000), r || '_osdba'
   from generate_series(1,1000000) as r;
```

在我的MAC电脑中生成100万数据用了4.5秒左右，如下所示：

```
osdba=# insert into tbl_user(id,user_id, user_name) select r,round(random()*2000000), r || '_osdba' from generate_series(1,1000000) as r;
INSERT 0 1000000
Time: 4548.128 ms (00:04.548)
```

然后我们用下面的SQL把数据生成到json的表tbl_user_jsonb中:

```
insert into tbl_user_jsonb(user_info) select row_to_json(tbl_user)::jsonb from tbl_user;
```

因为上面这条SQL又读又要写，慢一些，花了14秒左右：

```
osdba=# insert into tbl_user_jsonb(user_info) select row_to_json(tbl_user)::jsonb from tbl_user;
INSERT 0 1000000
Time: 14070.349 ms (00:14.070)
```

3. 建GIN索引演示索引的模式自由

建GIN索引的SQL如下：

```
create index idx_gin_tbl_user_jsonb_user_Info on tbl_user_jsonb using gin(user_Info);
```

注意建GIN索引的语法是 `using gin(<字段名>)` ，建此索引花了53秒，如下所示：

```
osdba=# create index idx_gin_tbl_user_jsonb_user_Info on tbl_user_jsonb using gin(user_Info);
CREATE INDEX
Time: 53702.887 ms (00:53.703)
```

我们这张表tbl_user_jsonb的数据如下：

```
osdba=# select * from tbl_user_jsonb limit 5;
 id |                                                  user_info
----+-------------------------------------------------------------------------------------------------------
  1 | {"id": 1, "user_id": 1650327, "user_name": "1_osdba", "create_time": "2018-12-12T16:15:57.681605+08:00"}
  2 | {"id": 2, "user_id": 986066, "user_name": "2_osdba", "create_time": "2018-12-12T16:15:57.681747+08:00"}
  3 | {"id": 3, "user_id": 1966018, "user_name": "3_osdba", "create_time": "2018-12-12T16:15:57.681755+08:00"}
  4 | {"id": 4, "user_id": 1143406, "user_name": "4_osdba", "create_time": "2018-12-12T16:15:57.681759+08:00"}
  5 | {"id": 5, "user_id": 1754355, "user_name": "5_osdba", "create_time": "2018-12-12T16:15:57.681763+08:00"}
(5 rows)

Time: 2.777 ms
```

这时我们按json中的“user_name”来查询：

```
osdba=# select * from tbl_user_jsonb where user_info @> '{"user_name":"232_osdba"}';
 id  |                                                  user_info
-----+-------------------------------------------------------------------------------------------------------
 232 | {"id": 232, "user_id": 597495, "user_name": "232_osdba", "create_time": "2018-12-12T16:15:57.682529+08:00"}
(1 row)

Time: 0.779 ms
```

从上面可以看到在0.779ms内就查出来了，就可想象一定是走到了索引。

解释一下上面SQL语句 `user_info @> '{"user_name":"232_osdba"}'` 中的“@>”是一个PostgreSQL中的特别的运算符，意思是包含，GIN索引需要使用这种语法来查询。

这时我们还可以按时间来查询，发现也很快就查询出来了：

```
osdba=# select * from tbl_user_jsonb where user_info @> '{"create_time":"2018-12-12T16:15:57.682529+08:00"}';
 id  |                                                  user_info
-----+-------------------------------------------------------------------------------------------------------
 232 | {"id": 232, "user_id": 597495, "user_name": "232_osdba", "create_time": "2018-12-12T16:15:57.682529+08:00"}
(1 row)

Time: 0.778 ms
```

这时我们插入一行数据，在json数据中增加一个没有的字段“nick_name”，如下所示：

```
insert into tbl_user_jsonb(user_info) values('{"id":1000001, "user_id":232323, "user_name":"tangcheng", "nick_name":"osdba"}');
INSERT 0 1
```

这时我们再用这个“nick_name”这个字段来查询：

```
osdba=# select * from tbl_user_jsonb where user_info @> '{"nick_name":"osdba"}';
    id    |                                        user_info
---------+-------------------------------------------------------------------------------------
 1000001 | {"id": 1000001, "user_id": 232323, "nick_name": "osdba", "user_name": "tangcheng"}
(1 row)

Time: 0.544 ms
```

发现可以很快可以查询出来，说明走了索引。从这里就可以知道，在PostgreSQL中不需要知道字段名就可以建一个通用的GIN索引，后续插入这些之前没有的字段后，也能按这些字段来查询，这就是索引的模式自由。

当然想看是否真的走到的索引，在PostgreSQL可以看执行计划：

```
osdba=# explain analyze select * from tbl_user_jsonb where user_info @> '{"nick_name":"osdba"}';
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Bitmap Heap Scan on tbl_user_jsonb  (cost=55.75..3351.08 rows=1000 width=141) (actual time=0.054..0.055 rows=1 loops=1)
   Recheck Cond: (user_info @> '{"nick_name": "osdba"}'::jsonb)
   Heap Blocks: exact=1
   ->  Bitmap Index Scan on idx_gin_tbl_user_jsonb_user_info  (cost=0.00..55.50 rows=1000 width=0) (actual time=0.042..0.042 rows=1 loops=1)
         Index Cond: (user_info @> '{"nick_name": "osdba"}'::jsonb)
 Planning time: 0.106 ms
 Execution time: 0.100 ms
(7 rows)

Time: 0.672 ms
```

从上面的结果 `Bitmap Index Scan on idx_gin_tbl_user_jsonb_user_info` s可以看出，按“nick_name”来查询数据时走到了GIN索引“idx_gin_tbl_user_jsonb_user_info”。

4. 总结

在PostgreSQL中使用GIN索引实现了JSON数据的索引的模式自由，当然在PostgreSQL在JSON数据库也支持使用Bree索引，这是就不是模式自由了，下面按JSON中的字段“user_name”建一个普通Btree索引：

```
create index idx_user_infob_user_name on tbl_user_jsonb((user_info ->> 'user_name'));
```

注意上面的SQL中 `((user_info ->> 'user_name'))` 是双括号，单括号会的报错：

```
osdba=# create index idx_user_infob_user_name2 on tbl_user_jsonb(user_info ->> 'user_name'
);
ERROR:  syntax error at or near "->>"
LINE 1: ...user_infob_user_name2 on tbl_user_jsonb(user_info ->> 'user_...
                                                             ^
Time: 1.230 ms
```

这主要是取json数据中的子字段的运算符“->>”的限制。

基于某个字段建索引要一些，只需要22.892秒，如下所示：

osdba=# create index idx_user_infob_user_name on tbl_user_jsonb((user_info ->> ‘user_name’));
CREATE INDEX
Time: 22892.580 ms (00:22.893)

前面建GIN索引花了53.702秒，虽然建这个GIN索引时间慢，但这个索引是对所有的数据都索引了，当然要慢一些，但通常我们使用JSON数据，就是为了模式自由，为了更好的扩展性和灵活性，性能低一些也是可以接受的。