INDEXING

Indexes help the database server find specific rows much faster than it could do without indexes, therefore enhancing database performance.

CREATE INDEX

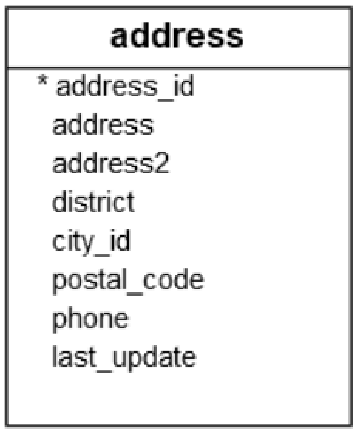
A simple version of CREATE INDEX statement is as follows:

```
CREATE INDEX index_name ON table_name [USING method]
(
    column_name [ASC | DESC] [NULLS {FIRST | LAST }],
    ...
);
```

The ASC and DESC specify the sort order. ASC is the default. NULLS FIRST or NULLS LAST specifies nulls sort before or after non-nulls. The NULLS FIRST is the default when DESC is specified and NULLS LAST is the default when DESC is not specified.

EXAMPLES

We will use the address table from the sample database for the demonstration.



The following query finds the address whose phone number is 223664661973:

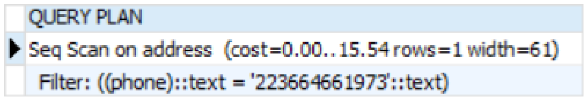
```
SELECT * FROM address
WHERE phone = '223664661973';
```

It is obvious that the database engine had to scan the whole address table to look for the address because there is no index available for the phone column.

To show the query plan, you use the EXPLAIN statement as follows:

```
EXPLAIN SELECT *
FROM address
WHERE phone = '223664661973';
```

Here is the output:



```
QUERY PLAN
Seq Scan on address  (cost=0.00..15.54 rows=1 width=61)
  Filter: ((phone)::text = '223664661973'::text)
```

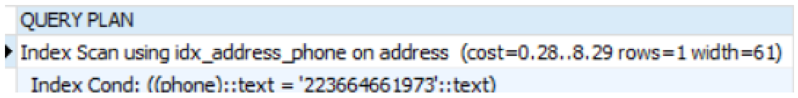
To create an index for the values in the phone column of the address table, you use the following statement:

```
CREATE INDEX idx_address_phone
ON address(phone);
```

Now, if you execute the query again, you will find that the database engine uses the index for lookup:

```
EXPLAIN SELECT *
FROM address
WHERE phone = '223664661973';
```

The following shows the output:



```
QUERY PLAN
Index Scan using idx_address_phone on address  (cost=0.28..8.29 rows=1 width=61)
  Index Cond: ((phone)::text = '223664661973'::text)
```

DROP INDEX

Sometimes, you may want to remove an existing index from the database system. To do it, you use the DROP INDEX statement as follows:

```
DROP INDEX  [ CONCURRENTLY]
[ IF EXISTS ]  index_name
[ CASCADE | RESTRICT ];
```

Note that you can drop multiple indexes at a time by separating the indexes by commas (,):

```
DROP INDEX index_name, index_name2,... ;
```

When you execute the DROP INDEX statement, PostgreSQL acquires an exclusive lock on the table and blocks other accesses until the index removal completes.

To force the command to wait until the conflicting transaction completes before removing the index, you can use the CONCURRENTLY option.

The DROP INDEX CONCURRENTLY has some limitations:

- First, the CASCADE option is not supported
- Second, executing in a transaction block is also not supported

Example we will drop the idx_address_phone index

DROP INDEX idx_address_phone;

DISPLAY INDEXES

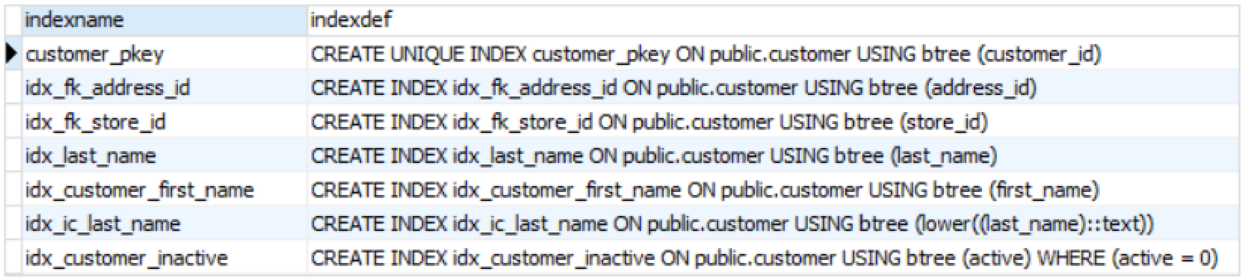
The pg_indexes view allows you to access useful information on each index in the PostgreSQL database. The pg_indexes view consists of five columns:

- schemaname: stores the name of the schema that contains tables and indexes.
- tablename: stores name of the table to which the index belongs.
- indexname: stores name of the index.
- tablespace: stores the name of the tablespace that contains indexes.
- indexdef: stores index definition command in the form of CREATE INDEX statement.

For example, to list all the indexes for the customer table, you use the following statement:

```
SELECT
    indexname,
    indexdef
FROM
    pg_indexes
WHERE
    tablename = 'customer';
```

Here is the output:



indexname	indexdef
customer_pkey	CREATE UNIQUE INDEX customer_pkey ON public.customer USING btree (customer_id)
idx_fk_address_id	CREATE INDEX idx_fk_address_id ON public.customer USING btree (address_id)
idx_fk_store_id	CREATE INDEX idx_fk_store_id ON public.customer USING btree (store_id)
idx_last_name	CREATE INDEX idx_last_name ON public.customer USING btree (last_name)
idx_customer_first_name	CREATE INDEX idx_customer_first_name ON public.customer USING btree (first_name)
idx_ic_last_name	CREATE INDEX idx_ic_last_name ON public.customer USING btree (lower((last_name)::text))
idx_customer_inactive	CREATE INDEX idx_customer_inactive ON public.customer USING btree (active) WHERE (active = 0)