Indexes on Tables

Let us go through the details related to indexes supported in RDBMS such as Postgres.

* An index can be unique or non unique.
* Unique Index - Data will be sorted in ascending order and uniqueness is enforced.
* Non Unique Index - Data will be sorted in ascending order and uniqueness is not enforced.
* Unless specified all indexes are of type B Tree.
* For sparsely populated columns, we tend to create B Tree indexes. B Tree indexes are the most commonly used ones.
* For densely populated columns such as gender, month etc with very few distinct values we can leverage bit map index. However bitmap indexes are not used quite extensively in typical web or mobile applications.
* Write operations will become relatively slow as data have to be managed in index as well as table.
* We need to be careful while creating indexes on the tables as write operations can become slow as more indexes are added to the table.
* Here are some of the criteria for creating indexes.
    * Create unique indexes when you want to enforce uniqueness. If you define unique constraint or primary key constraint, it will create unique index internally.
    * If we are performing joins between 2 tables based on a value, then the foreign key column in the child table should be indexed.
        * Typically as part of order management system, we tend to get all the order details for a given order using order id.
        * In our case we will be able to improve the query performance by adding index on **order_items.order_item_order_id**.
        * However, write operation will become a bit slow. But it is acceptable and required to create index on **order_items.order_item_order_id** as we write once and read many times over the life of the order.
* Let us perform tasks related to indexes.
    * Drop and recreate retail db tables.
    * Load data into retail db tables.
    * Compute statistics (Optional). It is typically taken care automatically by the schedules defined by DBAs.
    * Use code to randomly fetch 2000 orders and join with order_items - compute time.
    * Create index for order_items.order_item_order_id and compute statistics
    * Use code to randomly fetch 2000 orders and join with order_items - compute time.
* Script to create tables and load data in case there are no tables in retail database.

```
psql -U itversity_retail_user \
  -h localhost \
  -p 5432 \
  -d itversity_retail_db \
  -W

DROP TABLE order_items;
DROP TABLE orders;
DROP TABLE products;
DROP TABLE categories;
DROP TABLE departments;
DROP TABLE customers;

\i /data/retail_db/create_db_tables_pg.sql
\i /data/retail_db/load_db_tables_pg.sql
```

```
!pip install psycopg2
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: psycopg2 in /opt/anaconda3/envs/beakerx/lib/python3.6/site-
packages (2.8.6)
```

```
import psycopg2
```

```
%%time

from random import randrange
connection = psycopg2.connect(
    host='localhost',
    port='5432',
    database='itversity_retail_db',
    user='itversity_retail_user',
    password='retail_password'
)
cursor = connection.cursor()
query = '''SELECT *
FROM orders o JOIN order_items oi
    ON o.order_id = oi.order_item_order_id
WHERE o.order_id = %s
'''
ctr = 0
while True:
    if ctr == 2000:
        break
    order_id = randrange(1, 68883)
    cursor.execute(query, (order_id,))
    ctr += 1
cursor.close()
connection.close()
```

```
CPU times: user 73.8 ms, sys: 31.4 ms, total: 105 ms
Wall time: 19.6 s
```

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
%env
DATABASE_URL=postgresql://itversity_retail_user:retail_password@localhost:5432/itversity_retail
_db
```

```
env:
DATABASE_URL=postgresql://itversity_retail_user:retail_password@localhost:5432/itversity_reta
il_db
```

```
%%sql

CREATE INDEX order_items_oid_idx
ON order_items(order_item_order_id)
```

```
 * postgresql://itversity_retail_user:***@localhost:5432/itversity_retail_db
Done.
```

```
[]
```

```
%%time

from random import randrange
connection = psycopg2.connect(
    host='localhost',
    port='5432',
    database='itversity_retail_db',
    user='itversity_retail_user',
    password='retail_password'
)
cursor = connection.cursor()
query = '''SELECT *
FROM orders o JOIN order_items oi
    ON o.order_id = oi.order_item_order_id
WHERE o.order_id = %s
'''
ctr = 0
while True:
    if ctr == 2000:
        break
    order_id = randrange(1, 68883)
    cursor.execute(query, (order_id,))
    ctr += 1
cursor.close()
connection.close()
```

```
CPU times: user 49.1 ms, sys: 32.9 ms, total: 82 ms
Wall time: 265 ms
```

By Durga Gadiraju
© Copyright ITVersity, Inc.