How to create Index

1. Import data flights from `flights.sql`

2. Create an index

```
select * from flights where duration = 505;
```

```
create index fight_index on flight (duration);
```

EXPLAIN

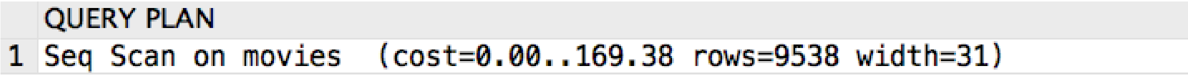
Part 1. Basic Search

Import `film.sql`

- Sequence Scan

```
explain select * from movies;
```

Result:



	QUERY PLAN
1	Seq Scan on movies (cost=0.00..169.38 rows=9538 width=31)

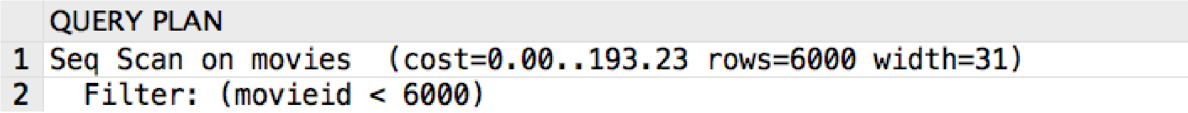
Total evaluation time cost = seq_page_cost + cpu_tuple_cost.

The time cost in seq is about 1.0, while time cost in cpu is about 0.01.
Suppose the data from 9538 rows in table movies are distributed in 74 disk page, and the total time cost would be **74 * 1.0 + 9538 * 0.01 = 169.38**

- Adding where condition

```
explain select * from movies where movieid<6000;
```

Result:

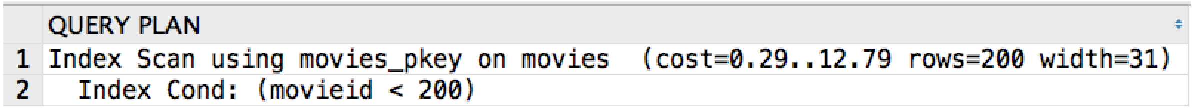


	QUERY PLAN
1	Seq Scan on movies (cost=0.00..193.23 rows=6000 width=31)
2	Filter: (movieid < 6000)

- Using index in where condition

```
explain select * from movies where movieid<200;
```

Result:



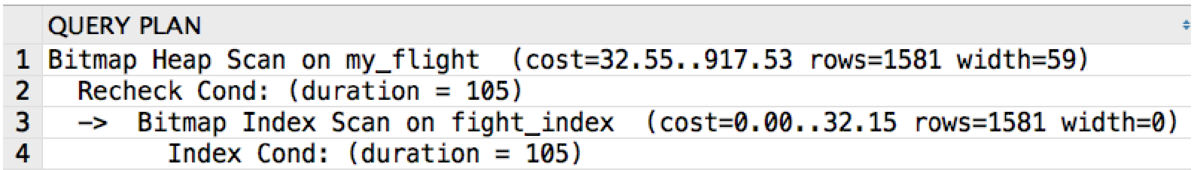
	QUERY PLAN
1	Index Scan using movies_pkey on movies (cost=0.29..12.79 rows=200 width=31)
2	Index Cond: (movieid < 200)

Part 2. Bitmap

- Bitmap Index scan

```
explain select * from flights where duration = 105;
```

Result:



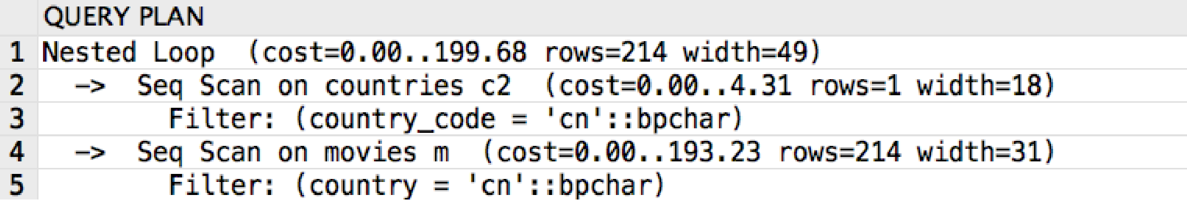
```
QUERY PLAN
Bitmap Heap Scan on my_flight  (cost=32.55..917.53 rows=1581 width=59)
  Recheck Cond: (duration = 105)
  ->  Bitmap Index Scan on fight_index  (cost=0.00..32.15 rows=1581 width=0)
        Index Cond: (duration = 105)
```

Part 3. Join

- Nest Loop

```
explain select *
        from movies m
          join
          countries c2 on m.country = c2.country_code
        where c2.country_code = 'cn';
```

Result:

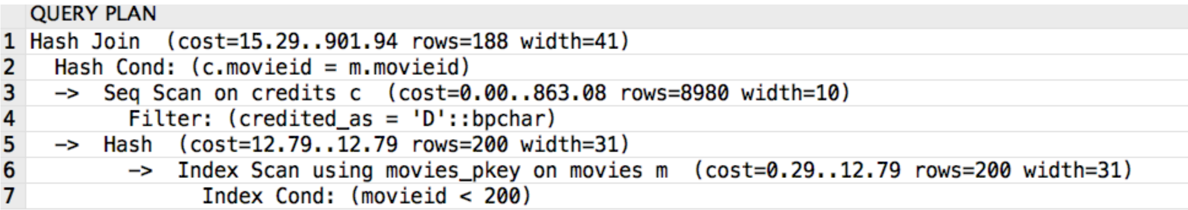


```
QUERY PLAN
Nested Loop  (cost=0.00..199.68 rows=214 width=49)
  ->  Seq Scan on countries c2  (cost=0.00..4.31 rows=1 width=18)
        Filter: (country_code = 'cn'::bpchar)
  ->  Seq Scan on movies m  (cost=0.00..193.23 rows=214 width=31)
        Filter: (country = 'cn'::bpchar)
```

- Hash Join

```
explain select *
        from movies m
          join credits c
            on m.movieid = c.movieid
        where c.credited_as = 'D' and m.movieid < 200;
```

Result:



```
QUERY PLAN
Hash Join  (cost=15.29..901.94 rows=188 width=41)
  Hash Cond: (c.movieid = m.movieid)
  ->  Seq Scan on credits c  (cost=0.00..863.08 rows=8980 width=10)
        Filter: (credited_as = 'D'::bpchar)
  ->  Hash  (cost=12.79..12.79 rows=200 width=31)
        ->  Index Scan using movies_pkey on movies m  (cost=0.29..12.79 rows=200 width=31)
              Index Cond: (movieid < 200)
```

Part 4. multi-key indexes

Step 1

```
create table my_movies
  as
    select *
   from movies;
```

Step 2

```
create index movies_multi_index on my_movies(movieid,year_released, runtime);
```

Step 3 Compare with following queries

```
explain select * from my_movies where movieid=20;
explain select * from my_movies where movieid<100 and year_released=2000;
explain select * from my_movies where year_released=2000;
explain select * from my_movies where runtime=200;
```

Part 5. Create index on function

Step 1

```
create index movies_title_index on my_movies(title);
```

Step 2 Compare following two queries, whether index is effected.

```
explain select * from my_movies where title = 'Armaan';
explain select * from my_movies where upper(title) = 'ARMAAN';
```

Step 3 Create index on upper() function

```
create index movies_upper_title_index on my_movies(upper(title));
```

Step 4 whether index is effected in follow query

```
explain select * from my_movies where upper(title) = 'ARMAAN';
```