Homework Assignment 5, by 22000546 Yeeun Lee

1.

(a) index

(b) ordered index

(c) clustering index

(d) sparse index

2.

(a)

Views, Indexes, Base tables

(b)

Performance degrades as the file grows.

3.

(a)

name	total_funding
Moderna	2700000000
Medtronic	367000000

(b)

call count_cpn_proc('Genetics', @tmp);

(c)

name
Moderna
Pacific Biosciences
Medtronic

4.

(a)

When data size is large, using dense indexes may require more space and may impose more maintenance overhead for insertion and deletion.

(b)

Clustering index is the case in which the sort order of a search key matches the sort order of a relation. Secondary index is same with non-clustering index. It is the case in which the sort order

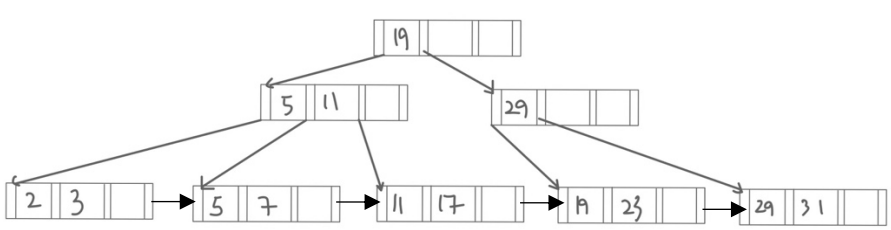
of a search key does not match the sort order of a relation.

(c)

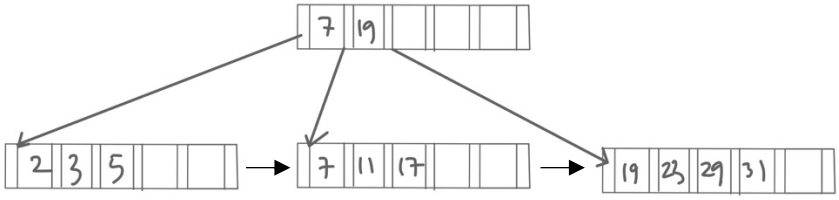
This is because as the index increases, the size of the capacity to contain the index increases, and the number of indexes that need to be updated when the operation is performed increases, increasing the overhead.

(d)

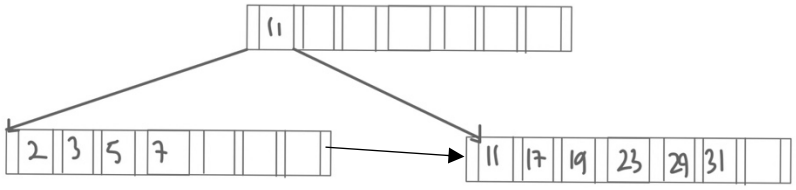
a.



b.

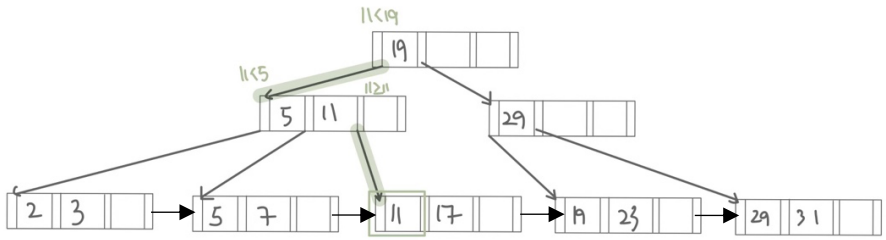


c.



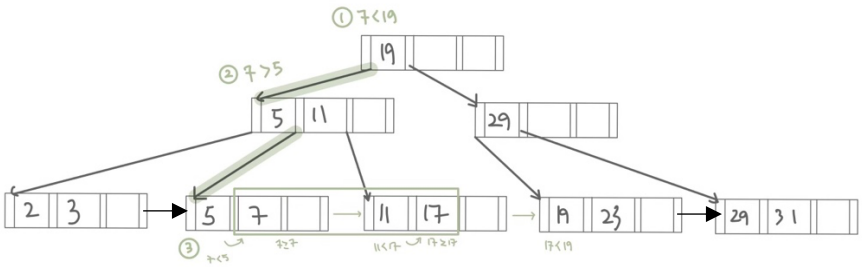
(e)

a.



1) After comparing the values of 19 and 11, P1 is selected because 11 is less than 19. 2) 11 is greater than 5, so look at the following value. Select P3 because 11 and 11 are the same. 3) 11 is equal to 11, so we have reached the value we want to find.

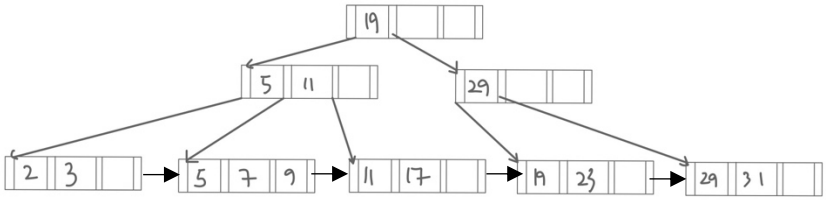
b.



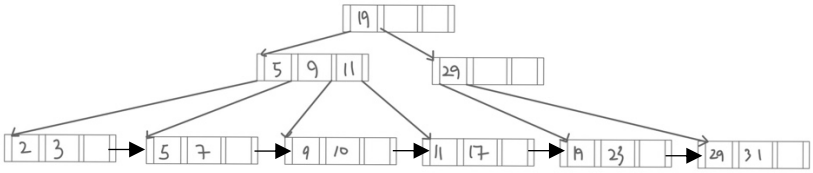
1) After comparing the values of 19 and 7, P1 is selected because 7 is less than 19. 2) 7 is greater than 5, so look at the following value. Select P2 because 7 is greater than 5 and smaller than 11. 3) 5 is less than 7, so look at the following value. 7 is equal to 7, so we have reached the value we want to find. Move on to the next values. 11, 17 is in our range.

(f)

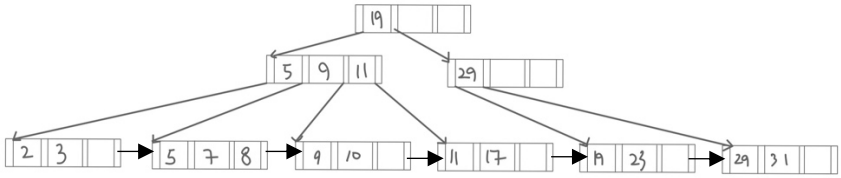
a-1. Insert 9



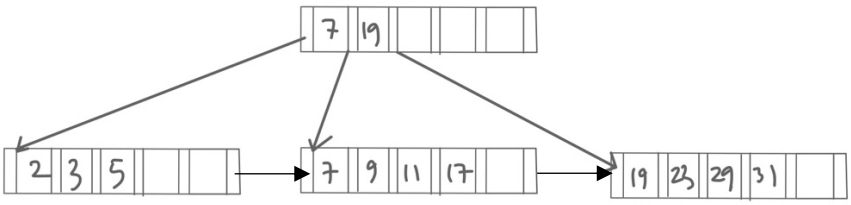
a-2. Insert 10



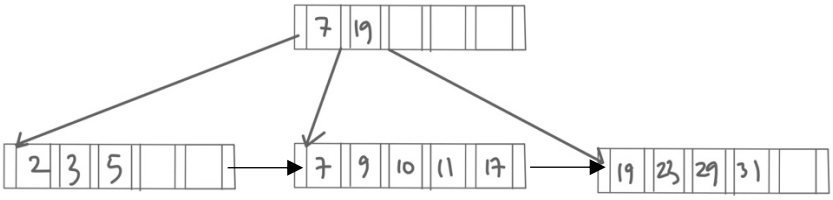
a-3. Insert 8



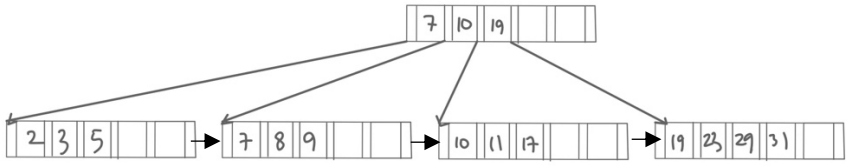
b-1. Insert 9



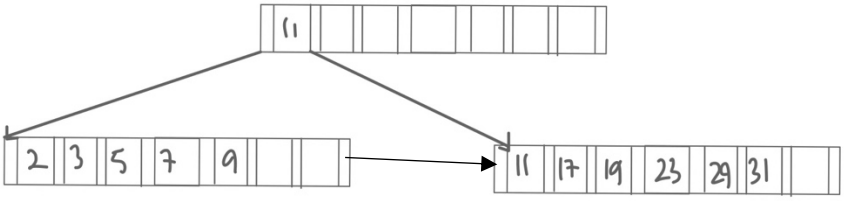
b-2. Insert 10



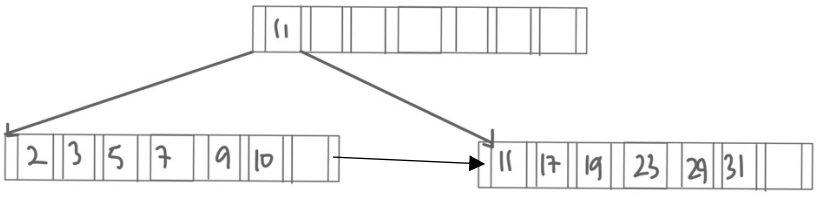
b-3. Insert 8



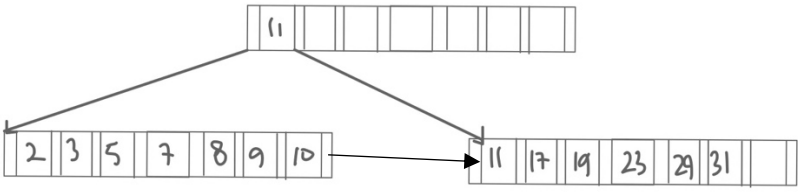
c-1. Insert 9



c-2. Insert 10



c-3. Insert 8



5.

(a)

location
MA
CA
NJ
MN
England
TN
MI

(b)

label	CNT
Wigwam	1
Warner Bros	1

(c)

name
Pat Metheny
John Scofield
Kenny Garrett

(d)

name	location
Marcus Miller	NY
John Scofield	NY

(e)

location	CNT
England	1
CA	2
MN	2
NJ	3
MA	4
NY	5

6.

(a)

inventory_id	row_no	rental_id	cust...	rental_date	PREV_RENTAL
1	1	11433	518	2005-08-02 20:13:10	NULL
1	2	14714	279	2005-08-21 21:27:43	2005-08-02 20:13:10
2	1	15453	359	2005-08-23 01:01:01	NULL
3	1	15421	541	2005-08-22 23:56:37	NULL
4	1	10883	301	2005-08-02 00:47:19	NULL
4	2	14624	344	2005-08-21 18:32:42	2005-08-02 00:47:19

Partitions are divided by inventory_id, sorted by rental_date. This query shows inventory_id, row number in partition, rental_id, customer_id, rental_date and the date of the previous record in the same partition, prev_rental.

(b)

payment_id	customer_id	rental_id	payment_date	amount	DAILY_SUM
1	1	76	2005-05-25 11:30:37	2.99	2.99
33	2	320	2005-05-27 00:09:24	4.99	6.98
60	3	435	2005-05-27 17:17:09	1.99	7.97
2	1	573	2005-05-28 10:35:23	0.99	3.97
108	5	731	2005-05-29 07:25:16	0.99	4.97
61	3	830	2005-05-29 22:43:55	2.99	3.98
109	5	1085	2005-05-31 11:15:43	6.99	8.98
110	5	1142	2005-05-31 19:46:38	1.99	8.98

This shows the payment_id, customer_id, rental_id, payment_date, amount, and total amount for 0.5 days before and after that, that is, for a daily sum by people whose customer_id is less or equal than 5, in May 2005.

(c)

```
select payment.customer_id, first_name, last_name, sum(amount)
from payment join customer on payment.customer_id = customer.customer_id
group by customer_id order by sum(amount) desc limit 5;
```

customer_id	first_name	last_name	sum(amount)
526	KARL	SEAL	221.55
148	ELEANOR	HUNT	216.54
144	CLARA	SHAW	195.58
178	MARION	SNYDER	194.61
137	RHONDA	KENNEDY	194.61

(d)

```
select rental.customer_id, first_name, last_name, sum(rental_id)
from rental join customer on rental.customer_id = customer.customer_id
group by customer_id order by sum(rental_id) desc limit 5;
```

cust...	first_name	last_name	sum(rental_id)
148	ELEANOR	HUNT	420168
144	CLARA	SHAW	344217
137	RHONDA	KENNEDY	344045
410	CURTIS	IRBY	342484
526	KARL	SEAL	337306

(e)

```
select film.film_id, title, sum(rental_id)
from (rental join inventory on rental.inventory_id = inventory.inventory_id)
join film on film.film_Id = inventory.inventory_id
group by film_id
order by sum(rental_id) desc
limit 7;
```

film_id	title	sum(rental_id)
210	DARKO DORADO	37471
468	INVASION CYCLONE	37050
547	MAGIC MALLRATS	36813
538	LOVERBOY ATTACKS	36746
491	JUMPING WRATH	36297
560	MARS ROMAN	36096
312	FIDDLER LOST	36022