SESSION #5

데이터베이스 객체 작성과 삭제 Part.2

By EDU Team 2nd (Jaehoon)

© 2018. SNU Growth Hackers all rights reserved

목차 소개

Part. 1

Chapter 01 데이터베이스 객체란?

Chapter 02 DDL 구문 (작성 · 삭제 · 변경 · 제약)

2.1 DDL구문 (작성)
2.2 DDL구문 (삭제)
2.3 DDL구문 (변경)
2.4 DDL구문 (제약)

Part. 2

Chapter 03 기본키 · 인덱스

3.1 기본키
3.2 인덱스 구조
3.3 인덱스 작성과 삭제

Chapter 04 뷰(View)

4.1 뷰 (view)
4.2 뷰 작성과 삭제

© 2018. SNU Growth Hackers all rights reserved

Chapter 3

Chapter 3.1 기본키

기본키에 대한 이해

기본키는 테이블의 행 한 개를 특정할 수 있는 검색 키이다.
기본키를 지정할 경우 인덱스도 같이 생성된다.

[기본키 제약 조건]

→ 기본키로 지정된 열은 NOT NULL 제약이 설정되어 있어야 한다.
→ 기본키로 지정된 열은 UNIQUE 해야 한다.

기본키 VS UNIQUE 인덱스 논쟁
http://devbab.tistory.com/6
http://tip.daum.net/question/62998497

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.1 기본키

기본키 제약 이해하기

```
mysql> use sample;
Database changed
mysql> CREATE TABLE sample634(
    -> p INTEGER NOT NULL,
    -> a VARCHAR(30),
    -> CONSTRAINT pkey_sample634 PRIMARY KEY(p)
    -> ) ;
Query OK, 0 rows affected (0.09 sec)

mysql> DESC sample634;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| p     | int(11)     | NO   | PRI | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

```
mysql> INSERT INTO sample634 VALUES(1, '첫째줄') ;
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO sample634 VALUES(2, '둘째줄') ;
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO sample634 VALUES(3, '셋째줄') ;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM sample634 ;
+---+--------+
| p | a      |
+---+--------+
| 1 | 첫째줄 |
| 2 | 둘째줄 |
| 3 | 셋째줄 |
+---+--------+
3 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.1 기본키

기본키 제약 이해하기 (INSERT의 경우)

```
INSERT INTO sample634 VALUES (2, '넷째줄') ;
```

〉〉〉 ERROR 1062 (23000) : Duplicate entry '2' for key 'PRIMARY'

해당 INSERT 명령은 기본키 제약에 위반되어 실행할 수 없다.
p열에는 이미 '2'라는 값이 존재하기 때문에 행을 추가할 경우 중복되는 값이 없어야 한다는 기본키 제약을 위반하게 된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.1 기본키

기본키 제약 이해하기 (UPDATE의 경우)

```
UPDATE sample634 SET p = 2 WHERE p = 3 ;
```

〉〉〉 ERROR 1062 (23000) : Duplicate entry '2' for key 'PRIMARY'

해당 UPDATE 명령은 기본키 제약에 위반되어 실행할 수 없다.
p열에 있는 값 '3'을 '2'로 바꾸게 될 경우 한 열 내에 중복되는 값이 발생하기 때문에 기본키 제약을 위반하게 된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.1 기본키

복수의 열로 기본키 구성하기

기본키를 구성하는 열은 복수라도 상관이 없다.
다만, 복수의 열을 기본키로 지정했을 경우,
키를 구성하는 모든 열을 사용해서 중복하는 값이 없어야 한다.

ex) …

a	b
1	1
1	2
2	1
2	2
3	1

a열과 b열이 각각 기본키로 지정되었다고 할 때,

1) a열만 보았을 때에는 중복되는 값이 존재하지만
2) b열이 다르다면 키 전체로서는 중복되지 않는다고 간주한다.
   → (a,b) ::: (1,1), (1,2), (2,1), (2,2)

따라서 기본키 제약에 위반되지 않는다. 하지만 만약 이 상태에서 키가 완전히 동일한 값으로 INSERT 명령을 실행하면 기본키 제약에 위반 된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

인덱스에 대한 이해

기본키 → **데이터 무결성**이 목적! & 테이블 간의 관계를 선언
인덱스 → **검색속도 향상**이 목적! & 반드시 UNIQUE하진 X

인덱스는 데이터베이스 객체 중 하나로서 **검색 속도를 향상하는 기능**을 가지고 있다.
(여기서 검색이란 SELECT 명령에 WHERE 구로 조건을 지정하고 그에 일치하는 행을 찾는 일련의 과정을 말한다.)

인덱스에는 검색 시에 쓰이는 키워드와 대응하는 데이터 행의 장소가 저장되어 있다.

일반적으로 테이블에 인덱스를 작성하면 테이블 데이터와는 별개로 인덱스용 데이터가 저장장치에 만들어진다. 하지만 인덱스는 테이블에 의존하는 객체이기 때문에 **테이블을 삭제할 경우 인덱스도 같이 삭제**된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

검색에 사용하는 알고리즘

1 풀 테이블 스캔 (full table scan)

인덱스가 지정되지 않은 테이블을 검색할 때 사용하는 알고리즘이다.
처리방법은 매우 단순한데, 테이블에 저장된 모든 값을 처음부터 차례로 조사해나가는 것이다. 따라서 행이 1000건이 있을 경우 최대 1000번까지 값을 비교하게 된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

검색에 사용하는 알고리즘

2 이진 탐색 (binary search)

이진 탐색은 처음부터 순서대로 조사하는 것이 아닌, 집합을 반으로 나누어가며 조사하는 검색 방법이다. 따라서 조사를 시작할 때에도 집합의 가운데에서부터 시작한다.

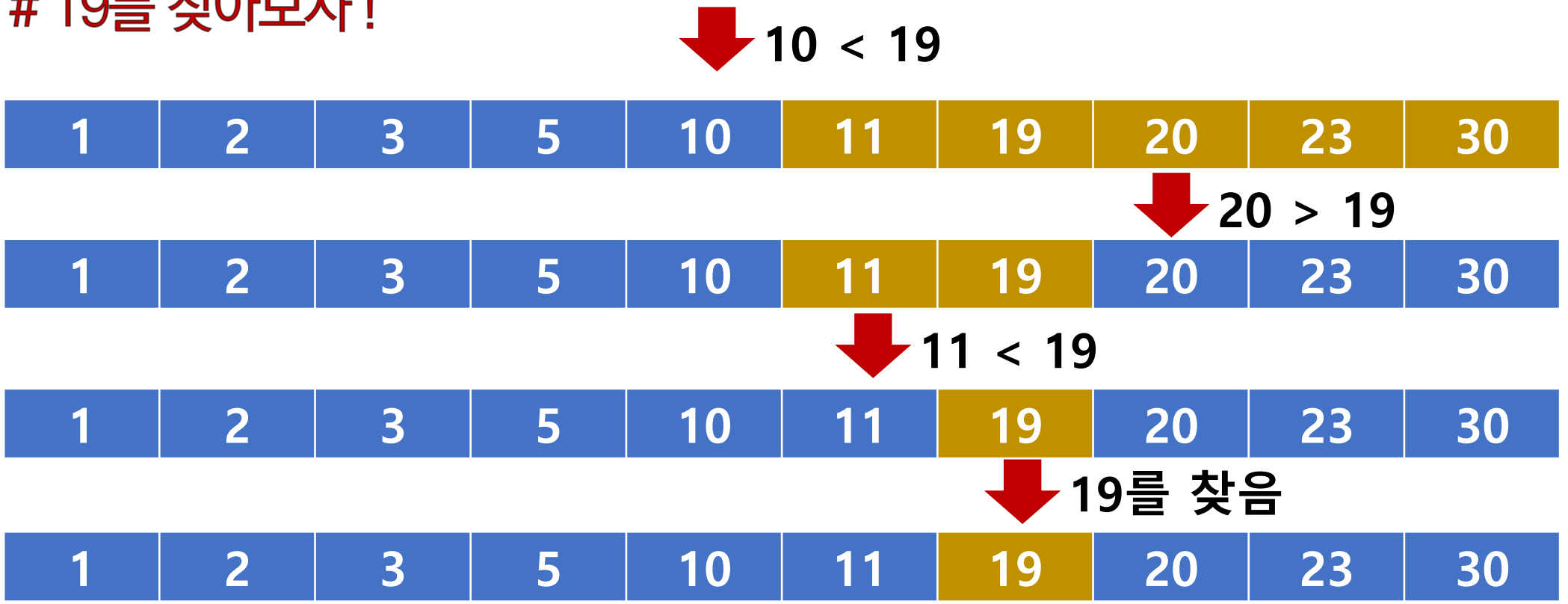
다음 장의 예시를 참고하여 이해를 하자.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

검색에 사용하는 알고리즘 – 이진 탐색 (binary search) 예시

19를 찾아보자 !



© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

검색에 사용하는 알고리즘

3 이진 트리 (binary tree)

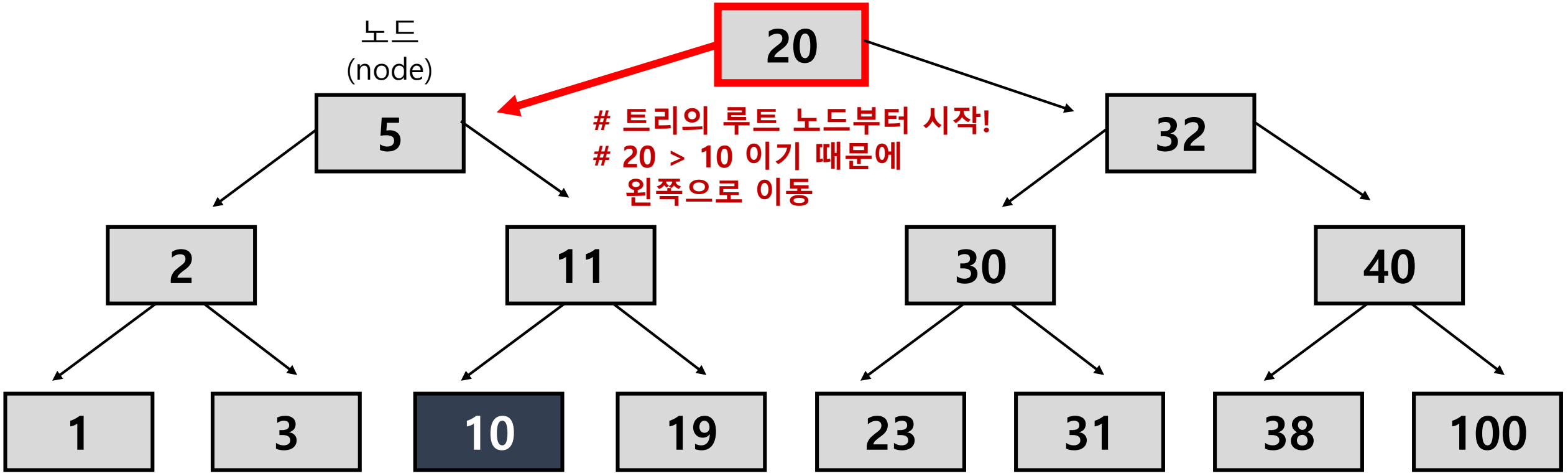
이진 탐색은 데이터가 미리 정렬되어 있어야 한다는 단점이 있다.
하지만 이진 트리는 정렬 여부와는 상관이 없다.
일반적으로 테이블에 인덱스를 작성하면 이진 트리라는 데이터 구조로 작성된다.

다음 장의 예시를 참고하여 이해를 하자.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

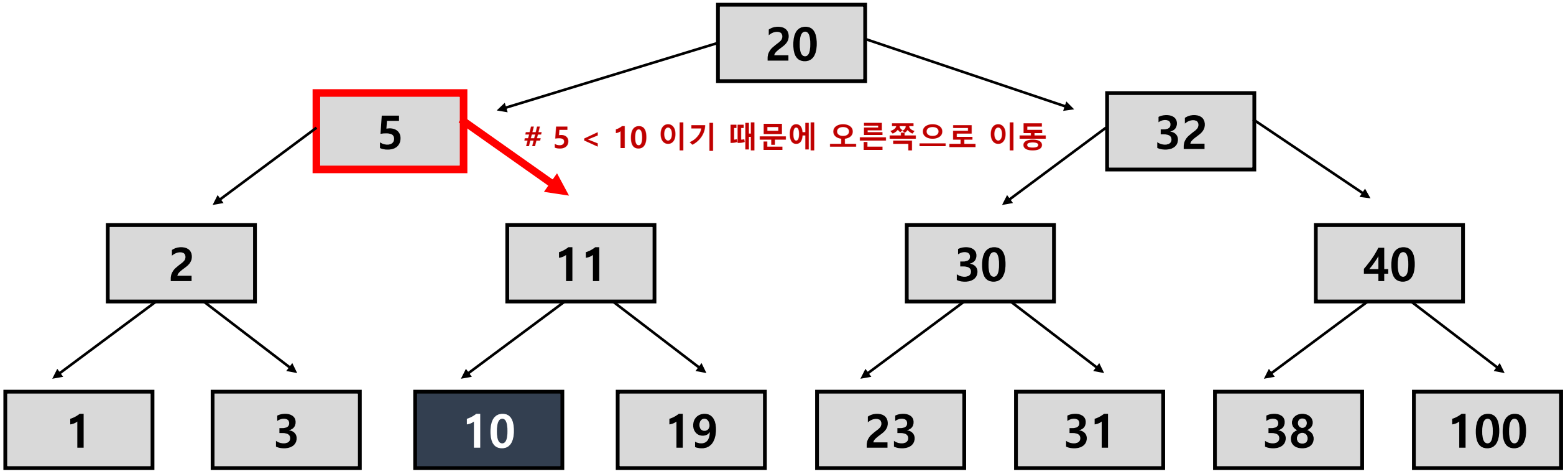
검색에 사용하는 알고리즘 – 이진 트리 (binary tree) 예시



© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

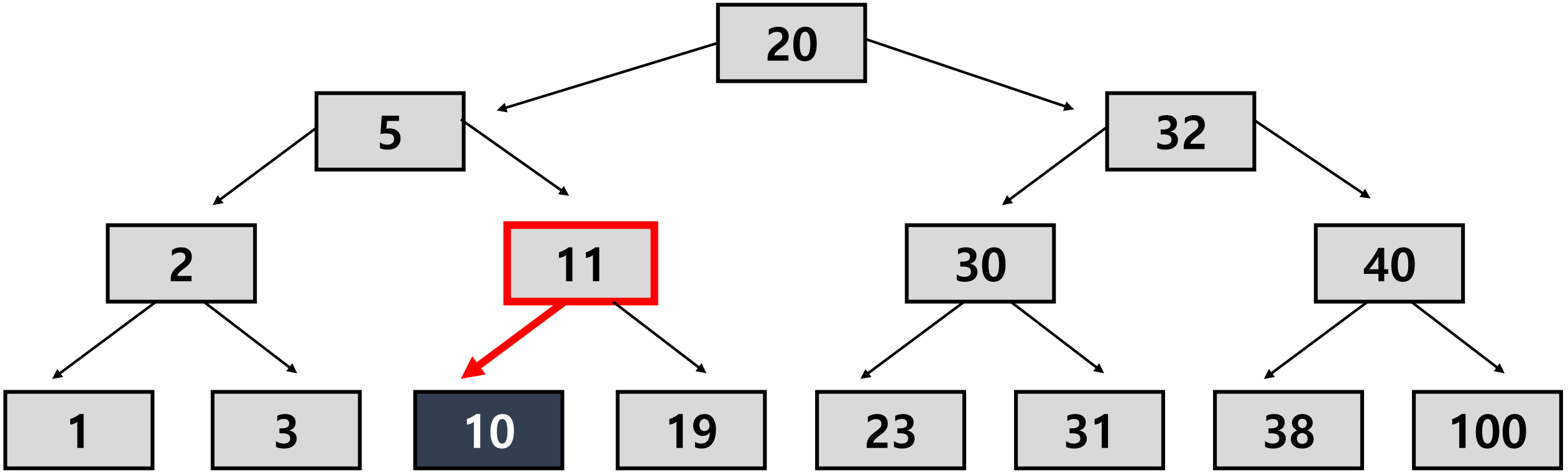
검색에 사용하는 알고리즘 – 이진 트리 (binary tree) 예시



© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

검색에 사용하는 알고리즘 – 이진 트리 (binary tree) 예시



11 > 10 이기 때문에 왼쪽으로 이동

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.2 인덱스 구조

유일성

만약 같은 값을 가지는 노드가 여러 개 있을 때의 이진 트리 결과에 대한 의문이 생길 수 있다.

사실 이진 트리에서는 집합 내 중복하는 값을 가질 수 없다.
노드의 가지는 큰 쪽과 작은 쪽의 두 가지로 나뉘며,
같은 값을 허용하기 위해서는 '같은'이라는 제 3의 가지를 가질 필요가 있다.
→ "이진"이 아니게 된다.

이진 트리에서 같은 값을 가지는 노드를 여러 개 만들 수 없다는 특성은
키에 대하여 유일성을 가지게 할 경우에만 유용하다. (UNIQUE + NOT NULL)
그래서 기본키 제약은 이진 트리로 인덱스를 작성하는 데이터베이스가 많다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 작성

```
CREATE INDEX isample65 ON sample62(no) ;
```

인덱스 명 테이블 명 칼럼 명

→ sample62 테이블의 no 칼럼에 isample65라는 이름의 인덱스를 지정한다.

테이블 크기에 따라 인덱스 작성시간도 달라진다.
행이 대량으로 존재하면 시간도 많이 걸리고 색인용 데이터를 저장하는 공간도 많이 필요하다.
MySQL에서 인덱스는 테이블 내의 객체가 되기 때문에 테이블 내에 이름이 중복되지 않도록 지정해 관리한다.

ⓒ 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 작성

```
mysql> DESC sample62 ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int(11)     | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

```
mysql> DESC sample62 ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int(11)     | NO   | MUL | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

동일한 값의 여러 항목이 허용되기 때문 (multiple)

```
mysql> CREATE INDEX isample65 ON sample62(no) ;
Query OK, 0 rows affected (0.79 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 작성

작성한 인덱스의 열을 WHERE 구로 조건을 지정하여 SELECT 명령으로 검색하면 처리속도가 향상된다.

INSERT 명령의 경우에는 인덱스를 최신 상태로 갱신하는 처리가 늘어나므로 처리속도가 조금 떨어진다.

isample65 라는 인덱스가 no열에 작성 되어있기 때문에
WHERE 구에 no열에 대한 조건식을 지정한 경우 SELECT 명령은 인덱스를 사용해 빠르게 검색할 수 있다.

그러나 WHERE 구의 조건식에 no열이 전혀 사용되지 않으면 SELECT 명령은 isample62 인덱스를 사용할 수 없다.

참고로 MySQL에서는 NULL 값도 인덱스에 들어갈 수 있다. (물론 권장사항은 아니다)

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 작성

EXPLAIN 구문 익히기 전에 데이터 추가해두기 ~

```
mysql> INSERT INTO sample62 VALUES(1,'python','2018-01-01'),(2,'java','2018-05-05'),(3,'c','2018-06-06') ;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM sample62 ;
+----+--------+------------+
| no | a      | b          |
+----+--------+------------+
|  1 | python | 2018-01-01 |
|  2 | java   | 2018-05-05 |
|  3 | c      | 2018-06-06 |
+----+--------+------------+
3 rows in set (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 작성

EXPLAIN 구문을 통해서 SQL 구문이 어떤 상태로 실행되는지 설명해준다.
이 경우에는 실제로 인덱스를 사용해 검색하는지의 여부를 확인할 수 있다.

EXPLAIN SELECT * FROM sample62 WHERE no = 1 ;

```
mysql> EXPLAIN SELECT * FROM sample62 WHERE no = 1 ;
+----+-------------+----------+------------+------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table    | partitions | type | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+----------+------------+------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | sample62 | NULL       | ref  | isample65     | isample65 | 4       | const |    1 |   100.00 | NULL  |
+----+-------------+----------+------------+------+---------------+---------+---------+-------+------+----------+-------+
1 row in set, 1 warning (0.00 sec)
```

EXPLAIN SELECT * FROM sample62 WHERE a = 'python' ;

```
mysql> EXPLAIN SELECT * FROM sample62 WHERE a = 'python' ;
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | sample62 | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    3 |    33.33 | Using where |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 삭제

```
DROP INDEX isample65 ON sample62 ;
```

인덱스 명 테이블 명

→ sample62 테이블의 인덱스를 삭제한다.

만약에 테이블 자체를 DROP한다면 역시 인덱스도 같이 DROP 된다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 3.3 인덱스 작성과 삭제

인덱스 삭제

```
mysql> DESC sample62 ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int(11)     | NO   | MUL | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> DESC sample62 ;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| no    | int(11)     | NO   |     | NULL    |       |
| a     | varchar(30) | YES  |     | NULL    |       |
| b     | date        | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

```
mysql> DROP INDEX isample65 ON sample62 ;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 4

Chapter 4.1 뷰(view)

뷰(view)에 대한 이해

본래 데이터베이스 객체로 등록할 수 없는 SELECT 명령을,
객체로서 이름을 붙여 관리할 수 있도록 한 것이 뷰(view)이다.

따라서 뷰를 참조하면 그에 정의된 SELECT 명령의 실행결과를 테이블처럼 사용할 수 있다.

뷰 역시 데이터베이스 객체 중 하나이다.
SELECT 명령으로 이루어지는 뷰는 테이블처럼 데이터를 쓰거나 지울 수 있는 저장공간을 가지지 않는다.
  이 때문에 테이블처럼 취급 할 수 있다고는 해도 SELECT 명령에서만 사용하는 것을 권장한다.
INSERT, UPDATE, DELETE 명령에서도 조건이 맞으면 가능하지만 사용할 때 주의할 필요가 있다.

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.1 뷰(view)

뷰(view)에 대한 이해

장점
1. 복잡한 쿼리를 단순화해서 사용할 수 있다. (특히 서브쿼리를 사용할 경우)
2. 쿼리를 재사용할 수 있다. (SELECT 문을 다시 쓰지 않고, 뷰 이름을 사용)

단점
1. ~~한 번 정의된 뷰는 변경할 수 없다. (근데 데이터 타입도 변경되고 이름도 변경되고…)~~
2. INSERT, DELETE, UPDATE** 작업에 많은 제한 사항을 가진다.
3. 뷰 자체만의 인덱스를 가질 수 없다.

**단점 2번 참고 (UPDATE 관련)
https://stackoverflow.com/questions/13793722/updating-views-in-mysql
http://recoveryman.tistory.com/181

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.2 뷰 작성과 삭제

뷰의 작성

VIEW 구문에서 쓰는 AS는 생략이 불가능하다.

```
CREATE VIEW sample_view_67 AS SELECT * FROM sample54 ;
```

뷰(view) 이름

뷰로 작성할 SELECT 구문

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.2 뷰 작성과 삭제

뷰의 작성

서브쿼리에 대한 별명을 안 붙이면 오류 발생함.

```
SELECT * FROM (SELECT * FROM sample54) sq ;
```

뷰를 작성하는 것으로
복잡한 SELECT 명령을 간략하게 표현할 수 있다!

```
SELECT * FROM sample_view_67 ;
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.2 뷰 작성과 삭제

뷰의 작성 (열을 지정하기)

```
CREATE VIEW sample_view_672(n, v, v2) AS
```

칼럼 이름

```
→ SELECT no, a, a*2 FROM sample54 ;
```

각각의 결과는 뷰의 칼럼 이름에 들어간다.

```
# 작성한 뷰를 한 번 사용해보자
SELECT * FROM sample_view_672 WHERE n = 1 ;
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.2 뷰 작성과 삭제

뷰의 작성 (열을 지정하기)

```
mysql> CREATE VIEW sample_view_672(n,v,v2) AS SELECT no,a,a*2 FROM sample54 ;
Query OK, 0 rows affected (0.05 sec)

mysql> SELECT * FROM sample_view_672 WHERE n = 1 ;
+------+------+------+
| n    | v    | v2   |
+------+------+------+
|    1 |  100 |  200 |
+------+------+------+
1 row in set (0.01 sec)
```

© 2018. SNU Growth Hackers all rights reserved

Chapter 4.2 뷰 작성과 삭제

뷰의 삭제

```
DROP VIEW sample_view_672 ;
```

뷰(view) 이름

```
mysql> DROP view sample_view_672 ;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM sample_view_672 WHERE n = 1 ;
ERROR 1146 (42S02): Table 'sample.sample_view_672' doesn't exist
```

© 2018. SNU Growth Hackers all rights reserved

Thank you

© 2018. SNU Growth Hackers all rights reserved