


Day24; 20220802(Quick Sort 난이도 上)

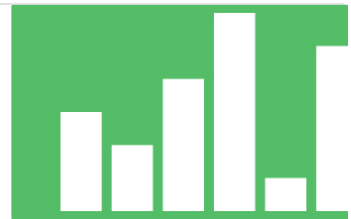
🕒 작성일시	@2022년 8월 2일 오전 9:18
📌 강의 번호	Lesson 24
📅 유형	@2022년 8월 2일
📎 자료	
☑ 복습	<input type="checkbox"/>
☰ 속성	

알고리즘을 애니메이션으로 구현하는 영상을 보여줍니다.

Sorting (Bubble, Selection, Insertion, Merge, Quick, Counting, Radix) - VisuAlgo


Sorting is a very classic problem of reordering items (that can be compared, e.g., integers, floating-point numbers, strings, etc) of an array (or a list) in a certain order (increasing, non-decreasing (increasing or flat), decreasing, non-increasing (decreasing or flat),

 <https://visualgo.net/en/sorting>



선생님 오픈 카톡

Austin님의 오픈프로필

 <https://open.kakao.com/o/sS6zZWNd>

 kakaotalk openchat



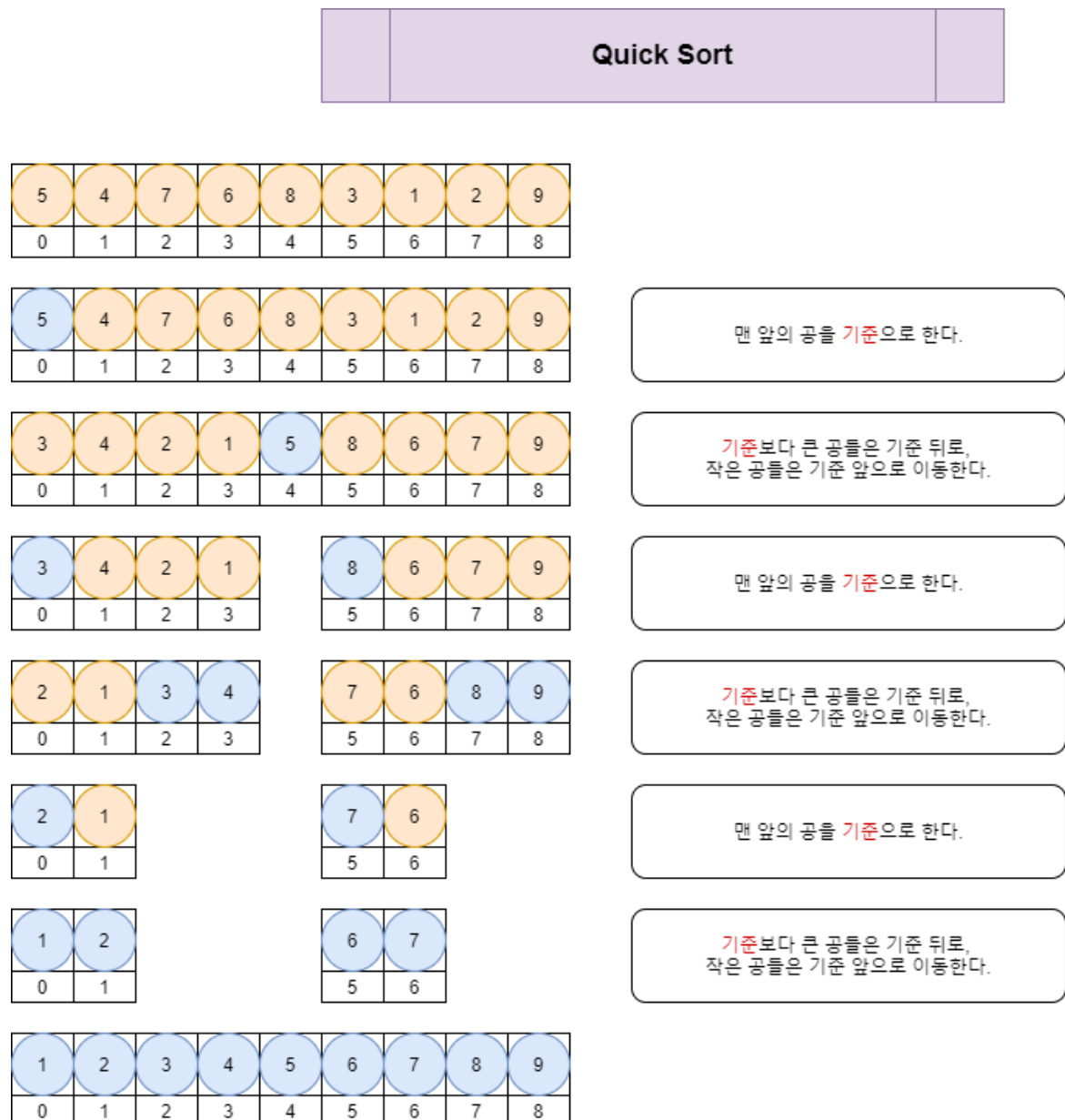
Quick Sort

데이터를 대소그룹 둘로 나누어 분해한 후에 전체를 최종적으로 정렬하는 방식의 알고리즘이다.

Divide and conquer(분할 정복법)

퀵 정렬은 대량의 데이터를 정렬할 때 매우 자주 사용된다. 유명한 알고리즘 중에서도 실제로 많이 사용되는 빈도가 가장 높고 중요한 알고리즘이다.

퀵 정렬은 '기준값을 선택한 후 그보다 작은 데이터 그룹과 큰 데이터 그룹으로 나눈다.'라는 처리를 반복 수행하여 데이터를 정렬하게 된다.



퀵 정렬의 알고리즘

퀵 정렬은 크게 2개의 처리로 구성된다.

1. 기준값을 경계로 데이터를 대소로 나누는 처리
2. 나눈 데이터에 대해 반복적으로 똑같은 작업을 실행

1. 기준값을 경계로 데이터를 대소로 나누는 처리

- 퀵 정렬의 핵심은 데이터를 대소로 나누는 처리이다.
- 배열의 왼쪽과 오른쪽부터 각각 변수를 움직여 대소로 정렬하자.

기준값보다 작은 공을 기준값의 앞으로 이동시키고 기준값보다 큰 공은 뒤로 이동시키는 것이 바로 퀵 정렬의 초석이 되는 처리이다.

배열 설정 : 먼저 배열을 준비하자. 정수형 배열로 이름은 arr로 요소수는 9개로 정한다. 따라서 첨자는 0 부터 8까지 사용된다.

arr

5	4	7	6	8	3	1	2	9
0	1	2	3	4	5	6	7	8

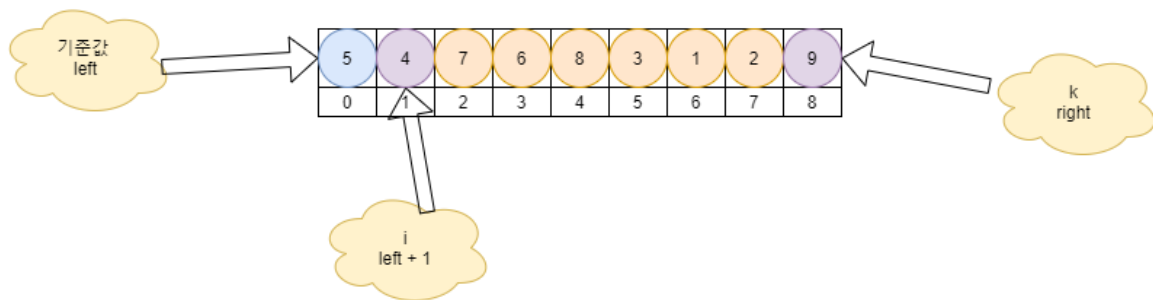
변수 설정

변수는 5개를 준비한다.

1. left - 정렬 범위에서 맨 앞 요소에 첨자를 넣는 변수
2. right - 정렬 범위에서 맨 끝 요소에 첨자를 넣는 변수
3. i - 기준값보다 큰 요소를 찾기 위한 변수
4. k - 기준값보다 작은 요소를 찾기 위한 변수
5. w - 데이터 교환용 임시 변수 temp

이 다섯개의 변수를 사용하여 우선 left와 right에 각각 정렬 범위 맨 앞 요소의 첨자와 마지막 요소의 첨자를 대입한다. 따라서 이번에는 (처음에는) left는 0, right은 8이 된다. 기준은 맨 앞 요소로 하기 때문에 arr[left]이 된다.

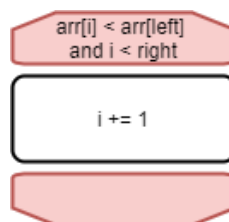
그리고 i에 left의 하나 오른쪽 left + 1로 정하고 k에는 right을 대입한다.



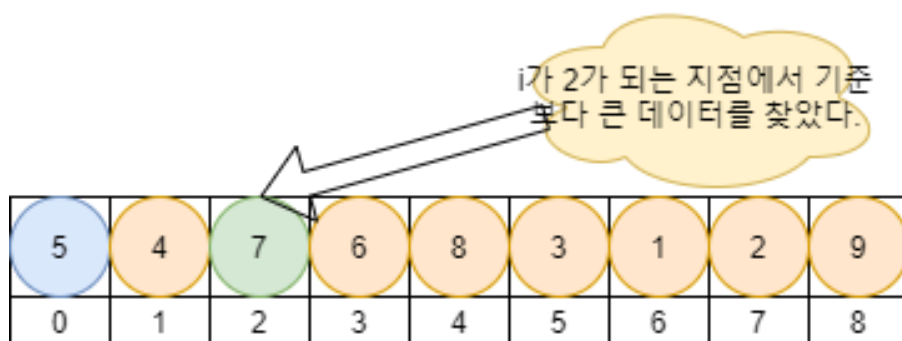
- 변수 i를 사용하여 기준값보다 큰 요소 찾기

i는 '기준값보다 큰 요소를 찾는 변수'이다. 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 있는지 확인하고 발견되면 그곳에서 멈춘다.

`arr[i] > arr[left] // arr[left]는 기준을 표시`



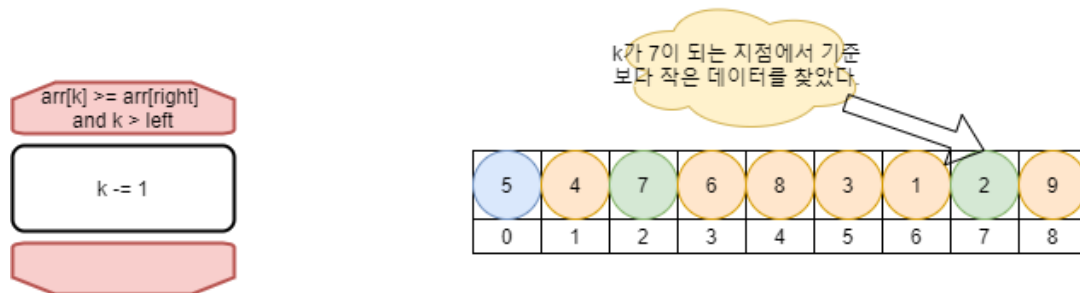
2가지 조건, 기준값 arr[left]보다 큰 값을 찾고 오른쪽 끝까지 찾지 못할 때까지 반복한다.



기준값보다 큰 요소를 발견했기 때문에 i는 일단 여기에서 멈춘다. 그리고 반대쪽 변수 k, 즉 작은 값 찾기로 넘어간다.

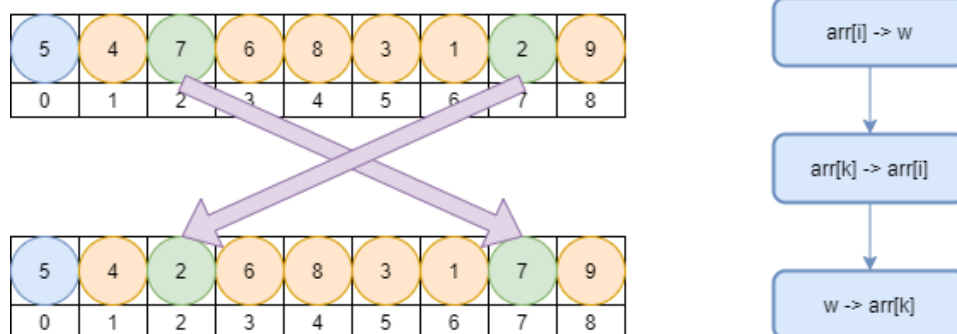
- 변수 k를 사용하여 기준값보다 작은 요소 찾기

k는 '기준값보다 작은 요소를 찾는 변수'이다. 현재 위치에서 하나씩 왼쪽으로 이동하면서 기준값보다 작은 요소가 있는지 확인하고 발견되면 그곳에서 멈춘다.



기준값보다 작은 요소를 발견했기 때문에 k도 일단 여기에서 멈춘다.

- 큰 데이터와 작은 데이터 교환하기



```

package quickSort;

import java.util.Arrays;

public class QuickSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int[] arr = {5,4,7,6,8,3,1,2,9};
        arr = quickSort(arr, 0, arr.length-1); // 어레이, 시작, 끝
        System.out.println(Arrays.toString(arr));
    }

    static int[] quickSort(int[] arr, int start, int end) {
        int p = partition(arr, start, end);
    }
    
```

```

    if(start < p-1) {
        quickSort(arr, start, p-1);
    }
    if(p < end) {
        quickSort(arr, p, end);
    }
    return arr;
}

static int partition(int[] arr, int start, int end) {
    int pivot = arr[(start + end) / 2];
    while(start <= end) {
        while(arr[start] < pivot) start++;
        while(arr[end] > pivot) end--;
        if(start <= end) {
            int tmp = arr[start];
            arr[start] = arr[end];
            arr[end] = tmp;
            start++;
            end--;
        }
    }
    return start;
}

}

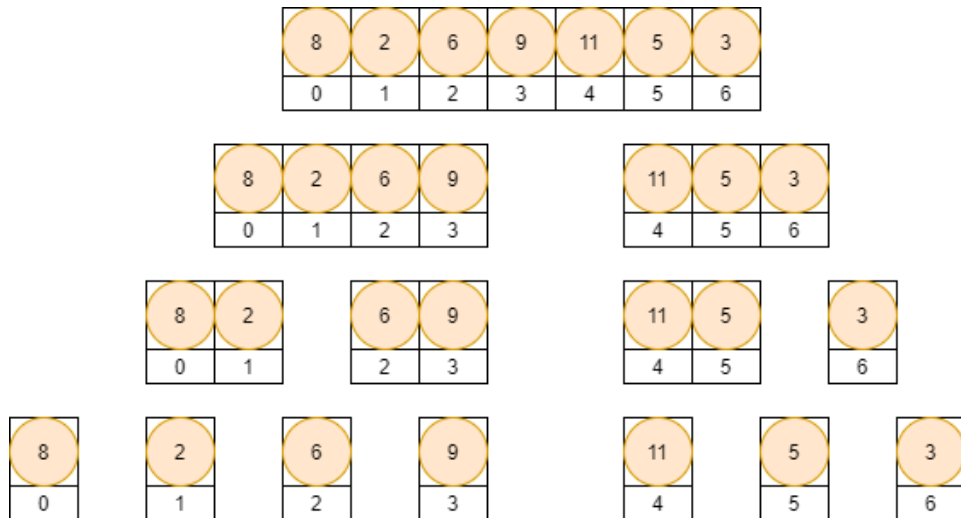
* 결과값은 [1, 2, 3, 4, 5, 6, 7, 8, 9] *

```

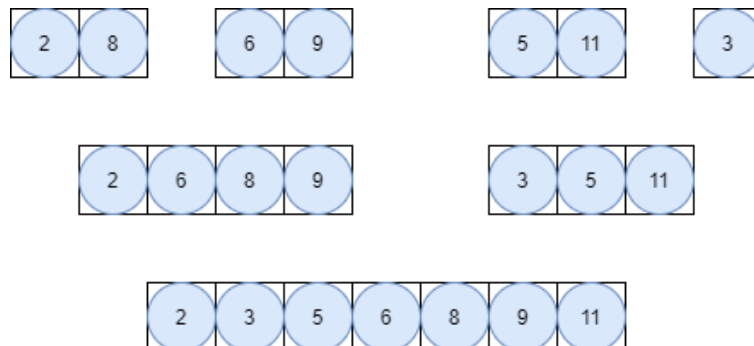
Merge Sort 병합 정렬

분할 정복 (Divide and Conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할한 후에 분할된 문제를 해결하고 그 결과를 다시 결합하는 알고리즘이다.

- 퀵 정렬이 제일 빠르고, 그 다음이 병합 정렬이 2번째로 빠르다.



먼저 데이터들을 정렬을 생각하지 않고 1개씩 될 때까지 나눈다. 나누는 방법은 배열을 반으로 쪼개고 그 쪼개진 배열을 또 다시 반으로 쪼개고 이 과정을 반복한다. 위에서 7개의 데이터를 분할하기 위해 이 과정을 3번 거쳤다.



분할이 완료된 후에는 데이터를 비교하면서 결합한다. 제일 앞에 있는 2와 8을 다시 합치는데 작은 수 2가 앞으로 큰 수 8이 뒤로 보낸 상태로 결합한다. 이 과정을 각각 2개씩 반복하여 합치고 그 합친 2개를 다시 합친다. 이때 각각의 그룹에서 먼저 제일 첫번째 값을 비교하여 작은 값을 추출한다. 그 다음 다시 한번 각각의 그룹의 제일 왼쪽 즉 작은 값을 또 다시 비교하여 둘 중 작은 값을 다시 추출한다. 이 과정을 반복하여 전체 정렬을 마치게 된다.

DATABASE(한글판 기준)

P103

Chapter 4 필터링

때로는 다음과 같이 테이블의 모든 행에서 작업할 수 있습니다.

- 새로운 데이터 웨어하우스 피드를 준비할 때 사용한 테이블에서 모든 데이터 제거
- 새 열이 추가된 후 테이블의 모든 행 수정
- 메시지 큐 테이블에서 모든 행 검색

4.1 조건 평가

(영문판, 한글판)

```
WHERE title = 'Teller' AND start_date < '2007-01-01'
```

```
WHERE first_name = 'STEVEN' AND create_date > '2006-01-01'
```

두 가지 조건을 모두 and 충족하는 타이틀이 텔러, 스티븐이고 동시에 날짜가 2007년, 2006년 1월 1일 이전인 행만 결과 포함됨. 조건이 여러개 있어도 AND연산자로 구분될 경우에는 결과셋에 모든 조건이 True인 경우만 포함된다. 즉 true로 평가된다.

4.1.1 괄호 사용

표 4-1 OR 조건을 사용한 두 개의 조건 평가

중간 결과	최종 결과
WHERE true OR true	true
WHERE true OR false	true
WHERE false OR true	true
WHERE false OR false	false

```
WHERE (first_name = 'STEVEN' OR last_name = 'YOUNG')  
AND create_date > '2006-01-01'
```


표 4-2 AND, OR 조건을 사용한 세 개의 조건 평가

중간 결과	최종 결과
WHERE (true OR true) AND true	true
WHERE (true OR false) AND true	true
WHERE (false OR true) AND true	true
WHERE (false OR false) AND true	false
WHERE (true OR true) AND false	false
WHERE (true OR false) AND false	false
WHERE (false OR true) AND false	false
WHERE (false OR false) AND false	false

3가지 조건이 있을 경우 최종 결과는 괄호 안의 2가지 조건의 결과와 최종 마지막 결과의 평가에 따라 최종 결과가 정해지게 된다.

4.1.2 not 연산자 사용

표 4-3 AND, OR, NOT 조건을 사용한 세 개의 조건 평가

중간 결과	최종 결과
WHERE NOT (true OR true) AND true	false
WHERE NOT (true OR false) AND true	false
WHERE NOT (false OR true) AND true	false
WHERE NOT (false OR false) AND true	true
WHERE NOT (true OR true) AND false	false
WHERE NOT (true OR false) AND false	false
WHERE NOT (false OR true) AND false	false
WHERE NOT (false OR false) AND false	false

not 연산자를 사용하여 평가 결과를 반대로 true의 경우는 false로 false의 경우에는 true 결과를 뒤집을 수 있다.

4.2 조건 작성

표현식은 아래의 내용들로 구성할 수 있다.

- 숫자
- 테이블 또는 뷰의 컬럼

- 'Maple Street'과 같은 문자열
- concat 과 같은 내장 함수들
- 서브 쿼리, 헤드텔러 등등과 같은 표현식 목록

조건의 유형

동등조건 - '열 = 값'

부등조건 - 두 표현식이 동일하지 않을 때 사용

```
mysql> SELECT c.email
      -> FROM customer c
      -> INNER JOIN rental r
      -> ON c.customer_id = r.customer_id
      -> WHERE date(r.rental_date) <> '2005-06-14';
```

```
+-----+
| email                                     |
+-----+
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| MARY.SMITH@sakilacustomer.org           |
| ...                                     |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
| AUSTIN.CINTRON@sakilacustomer.org       |
+-----+
16028 rows in set (0.03 sec)
```

동등조건을 사용한 데이터 변경

<영문판, 한글판 순서>

```
DELETE FROM account
WHERE status = 'CLOSED' AND YEAR(close_date) = 2002;
```

```
DELETE FROM rental
WHERE year(rental_date) = 2004;
```

4.3.2 범위 조건

```
mysql> SELECT emp_id, fname, lname, start_date
-> FROM employee
-> WHERE start_date < '2007-01-01';
```

```
SELECT emp_id, fname, lname, start_date
FROM employee
WHERE start_date < '2007-01-01';
```

이 쿼리는 2007년 1월 1일 이전의 모든 영화 대여 정보를 찾습니다.

```
mysql> SELECT customer_id, rental_date
-> FROM rental
-> WHERE rental_date < '2005-05-25';
```

customer_id	rental_date
130	2005-05-24 22:53:30
459	2005-05-24 22:54:33
408	2005-05-24 23:03:39
333	2005-05-24 23:04:41
222	2005-05-24 23:05:21
549	2005-05-24 23:08:07
269	2005-05-24 23:11:53
239	2005-05-24 23:31:46

```
8 rows in set (0.00 sec)
```

이 쿼리는 2005년 5월 25일 이전의 모든 영화 대여 정보를 찾습니다.

특정 범위내에 원하는 조건이 있는지를 확인하는 범위 조건을 작성할 수 있다.

이 유형은 보통 숫자 또는 시간데이터로 작업할 때 주로 발생한다.

between 연산자

<영문판, 한글판 순서>

범위의 상한과 하한이 모두 있을 때 사용한 between 연산자를 사용하여 하나의 조건으로 사용할 수 있다.

```
mysql> SELECT emp_id, fname, lname, start_date
-> FROM employee
-> WHERE start_date BETWEEN '2005-01-01' AND '2007-01-01';
```

```
mysql> SELECT account_id, product_cd, cust_id, avail_balance
-> FROM account
-> WHERE avail_balance BETWEEN 3000 AND 5000;
```

```
SELECT emp_id, fname, lname, start_date
FROM employee
WHERE start_date BETWEEN '2005-01-01' AND '2007-01-01';
```

```
SELECT account_id, product_cd, cust_id, avail_balance
FROM account
WHERE avail_balance BETWEEN 3000 AND 5000;
```

```
mysql> SELECT customer_id, rental_date
-> FROM rental
-> WHERE rental_date BETWEEN '2005-06-14' AND '2005-06-16';
```

customer_id	rental_date
416	2005-06-14 22:53:33
516	2005-06-14 22:55:13
239	2005-06-14 23:00:34
285	2005-06-14 23:07:08
310	2005-06-14 23:09:38
592	2005-06-14 23:12:46
...	
148	2005-06-15 23:20:26
237	2005-06-15 23:36:37
155	2005-06-15 23:55:27
341	2005-06-15 23:57:20
149	2005-06-15 23:58:53

364 rows in set (0.00 sec)

와일드 카드 사용

부분 문자열 일치를 검색할 때는 다음과 같은 사항이 궁금할 수 있습니다.

- 특정 문자로 시작·종료하는 문자열
- 부분 문자열(substring)로 시작·종료하는 문자열
- 문자열 내에 특정 문자를 포함하는 문자열
- 문자열 내에 부분 문자열을 포함하는 문자열
- 개별 문자에 관계없이 특정 형식의 문자열.

Table 4-4. Wildcard characters

Wildcard character	Matches
-	Exactly one character
%	Any number of characters (including 0)

표 4-4 와일드카드 문자

와일드카드 문자	일치
-	정확히 한 문자
%	개수에 상관없이 모든 문자(0 포함)

0 포함이라는 건 아무것도 안 와도 됨.

표 4-5 검색 표현식 사례

검색 표현식	해석
F%	F로 시작하는 문자열
%t	t로 끝나는 문자열
%bas%	문자열 'bas'를 포함하는 문자열
_ _t_	세 번째 위치에 t가 있는 4글자 문자열
_ _-_-_-_-_-	네 번째와 일곱 번째 위치에 -가 있는 11자리 문자열

ex1)

```
mysql> SELECT lname
-> FROM employee
-> WHERE lname LIKE '_a%e%';
```

lname
Barker
Hawthorne
Parker
Jameson

ex2)

```
mysql> SELECT cust_id, fed_id
-> FROM customer
-> WHERE fed_id LIKE '___-__-___';
```

cust_id	fed_id
1	111-11-1111
2	222-22-2222
3	333-33-3333
4	444-44-4444
5	555-55-5555

ex3)

```
mysql> SELECT emp_id, fname, lname
-> FROM employee
-> WHERE lname LIKE 'F%' OR lname LIKE 'G%';
```

emp_id	fname	lname
5	John	Gooding
6	Helen	Fleming
9	Jane	Grossman
17	Beth	Fowler

정규 표현식 사용

<한글판>

```
mysql> SELECT last_name, first_name
-> FROM customer
-> WHERE last_name REGEXP '^[QY]';
```

last_name	first_name
YOUNG	CYNTHIA
QUALLS	STEPHEN
QUINTANILLA	ROGER
YANEZ	LUIS
YEE	MARVIN
QUIGLEY	TROY

6 rows in set (0.16 sec)

4.4 Null

해당사항 없음, 아직 아려지지 않은 값, 정의되지 않은 값

Null로 작업할 때는 다음 사항들을 기억해야 합니다.

- Null일 수는 있지만, null과 같을 수는 없습니다.
An expression can be null, nut it can never equal null.
- 두 개의 null은 서로 같지 않습니다.
-

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
-> FROM employee
-> WHERE superior_emp_id IS NULL;
```

emp_id	fname	lname	superior_emp_id
1	Michael	Smith	NULL

1 row in set (0.00 sec)

```
mysql> SELECT rental_id, customer_id
-> FROM rental
-> WHERE return_date IS NULL;
```

```
+-----+-----+
| rental_id | customer_id |
+-----+-----+
|      11496 |          155 |
|      11541 |          335 |
|      11563 |           83 |
|      11577 |          219 |
|      11593 |           99 |
|      ...   |             |
|      15867 |          505 |
|      15875 |           41 |
|      15894 |          168 |
|      15966 |          374 |
+-----+-----+
183 rows in set (0.01 sec)
```

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
-> FROM employee
-> WHERE superior_emp_id IS NOT NULL;
```

```
SELECT emp_id, fname, lname, superior_emp_id
FROM employee
WHERE superior_emp_id IS NOT NULL;
```

Quiz 7

월급이 3000인 직원들의 이름, 월급, 직업을 출력해 보겠습니다.

ENAME	SAL	JOB
FORD	3000	ANALYST
SCOTT	3000	ANALYST


```
SELECT ENAME, SAL, JOB
FROM emp
WHERE sal >= 3000;
```

월급이 3000 이상인 직원들의 이름과 월급을 출력;

이름	월급
KING	5000
FORD	3000
SCOTT	3000

```
SELECT ENAME AS '이름',
SAL AS '월급'
FROM emp
WHERE sal >= 3000;
```

Quiz 8

이름이 SCOTT인 직원의 이름, 월급, 직업, 입사일, 부서 번호를 출력해 보겠습니다.

ENAME	SAL	JOB	HIREDATE	DEPTNO
SCOTT	3000	ANALYST	82/12/22	20

```
SELECT ename, sal, job, hiredate, deptno
FROM emp
WHERE ename = 'SCOTT';
```

Quiz 9

연봉이 36000 이상인 직원들의 이름과 연봉을 출력해 보겠습니다.

ENAME	연봉
KING	60000
FORD	36000
SCOTT	36000

```
SELECT ename,  
       sal * 12 AS '연봉'  
FROM emp  
WHERE sal * 12 >= 36000;
```

Quiz 10

월급이 1000에서 3000 사이인 직원들의 이름과 월급을 출력해 보겠습니다.

ENAME	SAL
BLAKE	2850
CLARK	2450
:	:
ADAMS	1100
MILLER	1300

```
SELECT ENAME, SAL
FROM emp
WHERE sal BETWEEN 1000 AND 3000;
```

Quiz 11

1982년도에 입사한 직원들의 이름과 입사일을 조회:

ENAME	HIREDATE
SCOTT	82/12/22
MILLER	82/01/11

```
SELECT ENAME, HIREDATE
FROM emp
WHERE HIREDATE BETWEEN '1982-01-01' AND '1982-12-31';
```

```
SELECT ENAME, HIREDATE
FROM emp
WHERE HIREDATE LIKE '1982-__-__';
```

Quiz 12

이름의 두 번째 철자가 M인 직원의 이름을 출력:

ENAME
SMITH

```
SELECT ENAME  
FROM emp  
WHERE ename LIKE '_M_';
```



이름이 A가 포함된 직원들을 전부 검색

1	SELECT	ENAME
2	FROM	emp
3	WHERE	ename LIKE '%A%';

emp (7r x 1c)	
ENAME	
BLAKE	
CLARK	
MARTIN	
ALLEN	
JAMES	
WARD	
ADAMS	

```
SELECT ENAME
FROM emp
WHERE ename LIKE '%A%';
```

Quiz 13

커미션이 NULL인 직원들의 이름과 커미션을 출력해 보겠습니다.

ENAME	COMM
KING	
:	:
MILLER	

1	SELECT ENAME, COMM
2	FROM emp
3	WHERE comm IS NULL;

emp (10r × 2c)	
ENAME	COMM
KING	(NULL)
BLAKE	(NULL)
CLARK	(NULL)
JONES	(NULL)
JAMES	(NULL)
FORD	(NULL)
SMITH	(NULL)
SCOTT	(NULL)

Quiz 14

직업이 SALESMAN, ANALYST, MANAGER인 직원들의 이름, 월급, 직업을 출력해 보겠습니다.

ENAME	SAL	JOB
BLAKE	2850	MANAGER
CLARK	2450	MANAGER
JONES	2975	MANAGER
MARTIN	1250	SALESMAN
ALLEN	1600	SALESMAN
TURNER	1500	SALESMAN
WARD	1250	SALESMAN
FORD	3000	ANALYST
SCOTT	3000	ANALYST

```

SELECT ENAME, SAL, JOB
FROM emp
WHERE (JOB = 'SALESMAN' or JOB = 'ANALYST' or JOB = 'MANAGER');

```

1	SELECT ENAME, SAL, JOB
2	FROM emp
3	WHERE JOB IN('SALESMAN', 'ANALYST', 'MANAGER');
4	
5	

emp (9r x 3c)		
ENAME	SAL	JOB
BLAKE	2,850	MANAGER
CLARK	2,450	MANAGER
JONES	2,975	MANAGER
MARTIN	1,250	SALESMAN
ALLEN	1,600	SALESMAN
TURNER	1,500	SALESMAN
WARD	1,250	SALESMAN
FORD	3,000	ANALYST

AND 연산자 진리 연산표

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR 연산자 진리 연산표

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT 연산자 진리 연산표

NOT	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL