

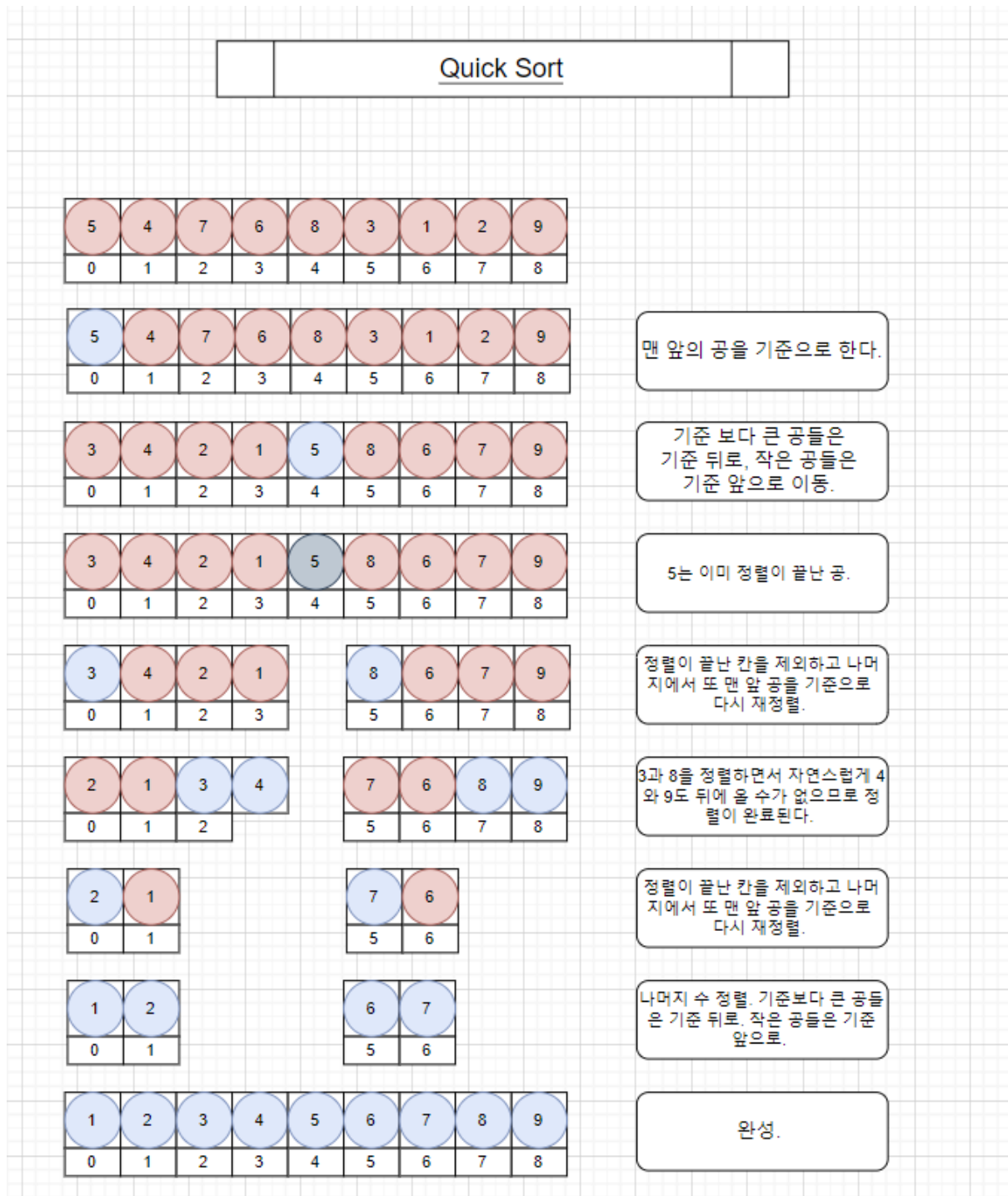
# Day25

🕒 작성일시	@2022년 8월 2일 오전 9:35
▼ 강의 번호	AUS-101
▼ 유형	
🔗 자료	
☑ 복습	<input type="checkbox"/>

## I . algorithm

### 1. Quick Sort

- 데이터를 대소그룹 둘로 나누어 분해한 후에 전체를 최종적으로 정렬하는 방식의 알고리즘.
- Divide and conquer (분할 정복법)
- 퀵정렬은 대량의 데이터를 정렬할때 매우 자주 사용한다. 유명한 알고리즘 중에서도 실제로 많이 사용되며, 빈도가 가장 높고 중요한 알고리즘이다.
- 퀵정렬은 '기준값을 선택한 후 그 보다 작은 데이터 그룹과 큰 데이터 그룹으로 나눈다.' 라는 처리를 반복 수행하여 데이터를 정렬하게 된다.



## (1) 퀵 정렬의 알고리즘

- 퀵 정렬은 크게 2개의 처리로 구성된다.

- ! 1) 기준 값을 경계로 데이터를 대소로 나누는 처리.
- 2) 나눈 데이터에 대해 반복적으로 똑같은 작업을 실행.

## 1) 기준 값을 경계로 데이터를 대소로 나누는 처리.

- 퀵정렬의 핵심은 데이터를 대소로 나누는 처리이다.
- 배열의 왼쪽과 오른쪽 부터 각각 변수를 움직여 대소를 정렬하자.

⇒ 기준 값보다 작은 공을 기준 값의 앞으로 이동시키고, 기준 값보다 큰 공은 뒤로 이동시키는 것이 바로 퀵정렬의 초석이 되는 처리이다.

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

### 1 - arr

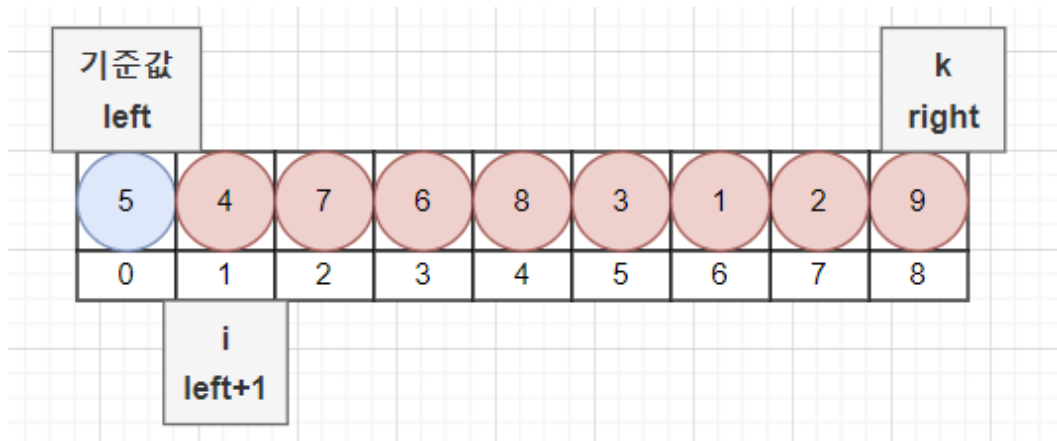
5	4	7	6	8	3	1	2	9
0	1	2	3	4	5	6	7	8

### 2 - 변수 설정

- 변수는 5개를 준비한다.
- left - 정렬 범위에서 맨 앞 요소에 첨자를 넣는 변수.
- right - 정렬 범위에서 맨 끝 요소에 첨자를 넣는 변수
- i - 기준값보다 큰 요소를 찾기 위한 변수.
- k - 기준 값보다 작은 요소를 찾기 위한 변수
- 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) left = 0, right = 8 이 된다.

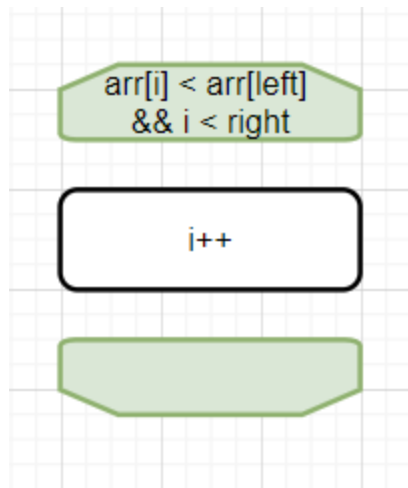
⇒ 기준은 맨 앞 요소로 하기 때문에 arr[left] 가 된다. 그리고 i에 left의 하나 오른쪽 left + 1로 정하고, k는 right을 대입한다.



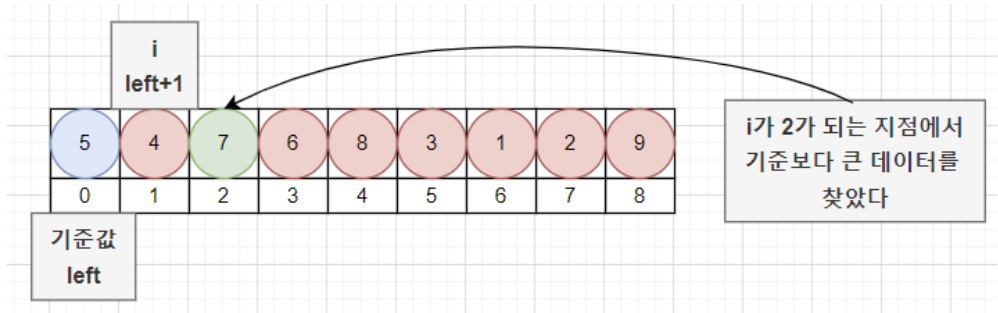
### 3 - 변수 i를 사용하여 기준 값보다 큰 요소 찾기.

- i는 기준값보다 큰 요소를 찾는 변수이다. 현재 위치에서 하나씩 오른쪽으로 이동하면서 기준값보다 큰 요소가 있는지 확인하고 발견되면 그 곳에서 멈춘다.

⇒  $arr[i] > arr[left]$  (기준).



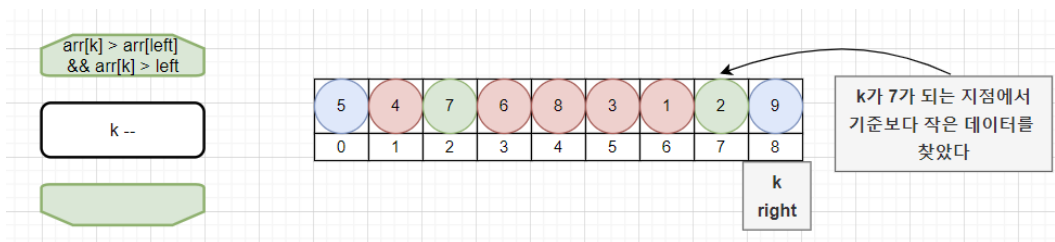
⇒ 반복의 두가지 조건. ①기준값( $arr[left]$ )보다 큰 값을 찾거나, ②오른쪽 끝까지 찾기 못할때까지 반복한다.



- 기준 값보다 큰 요소를 발견했기 때문에 i는 일단 여기서 멈춘다. 그리고 반대쪽 변수 k, 즉, 작은 값 찾기로 들어간다.

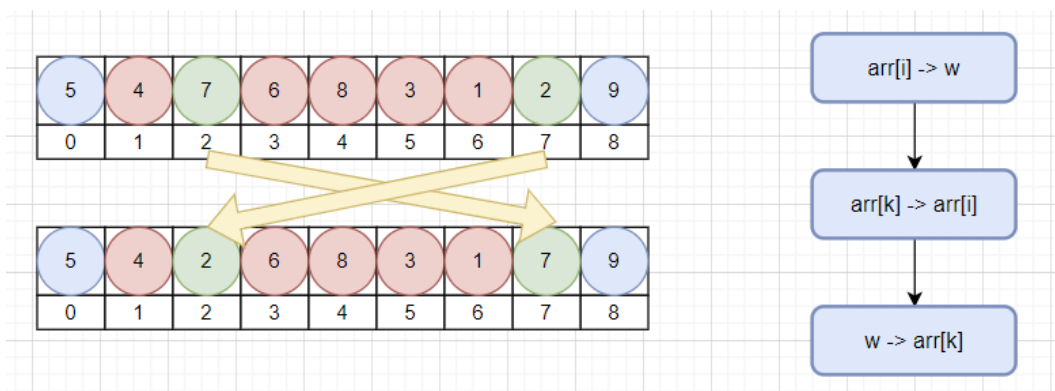
#### 4 - 변수 k를 사용하여 기준 값보다 작은 요소 찾기.

- k는 기준값보다 작은 요소를 찾는 변수이다. 현재 위치에서 하나씩 왼쪽으로 이동하면서 기준값보다 작은 요소가 있는지 확인하고 발견되면 그 곳에서 멈춘다

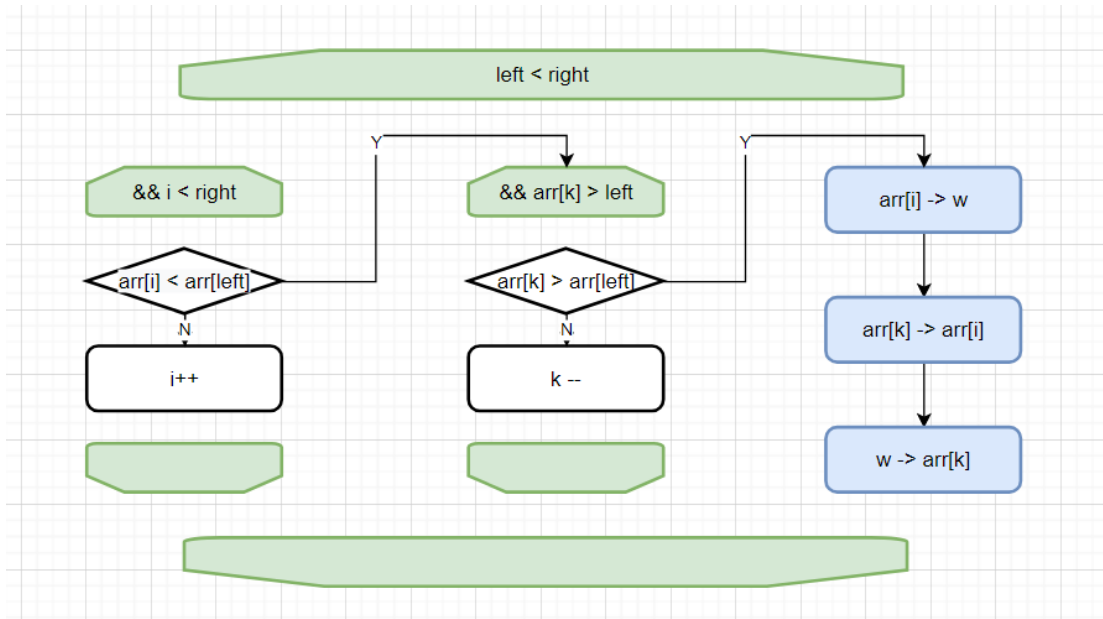


- 기준 값보다 작은 데이터를 찾으면 멈춘다.

#### 5 - 큰데이터와 작은 데이터를 교환하기.



## 6 - 반복문 생성



## 7 - 코드 생성 (쉬운 버전)

```

import java.util.Arrays;

public class Sort {

    public static void main(String[] args) {

        int[] arr = {5, 4, 7, 6, 8, 3, 1, 2, 9};
        arr = quickSort(arr,0, arr.length-1); //array, 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]; //기준값을 가운데로 잡는것.
        while(start <= end) {
            while(arr[start]<pivot) start ++;
            while(arr[end]>pivot) end--;
            if (start <= end) {

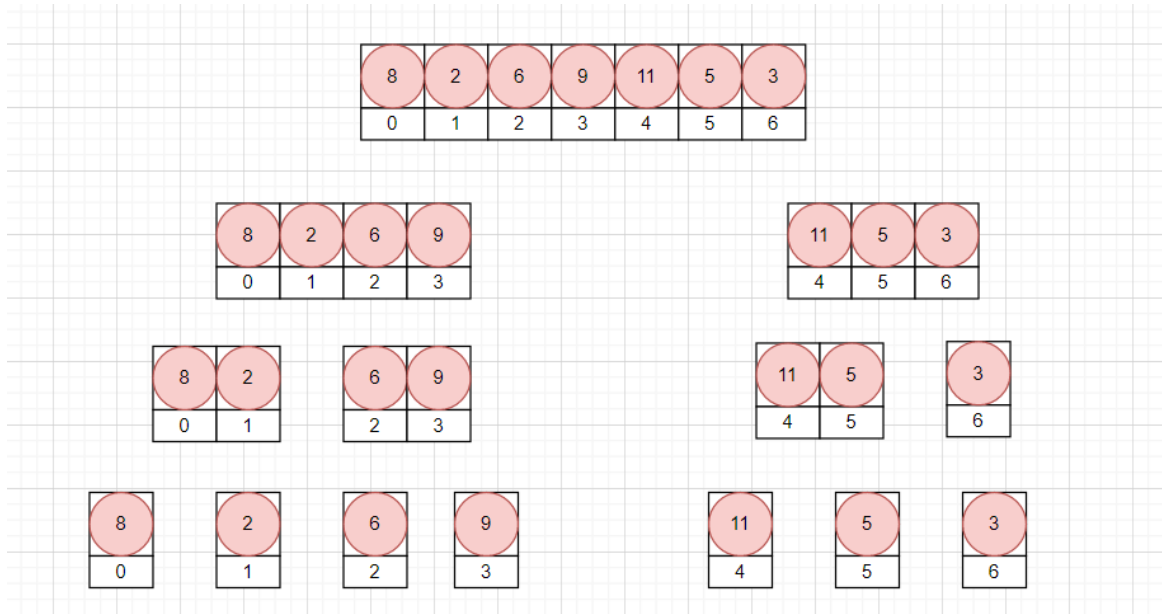
```

```
        int tmp = arr[start];
        arr[start] = arr[end];
        arr[end] = tmp;
        start++;
        end--;
    }
}
return start;
}
}
```

## 2. Merge Sort (병합 정렬)

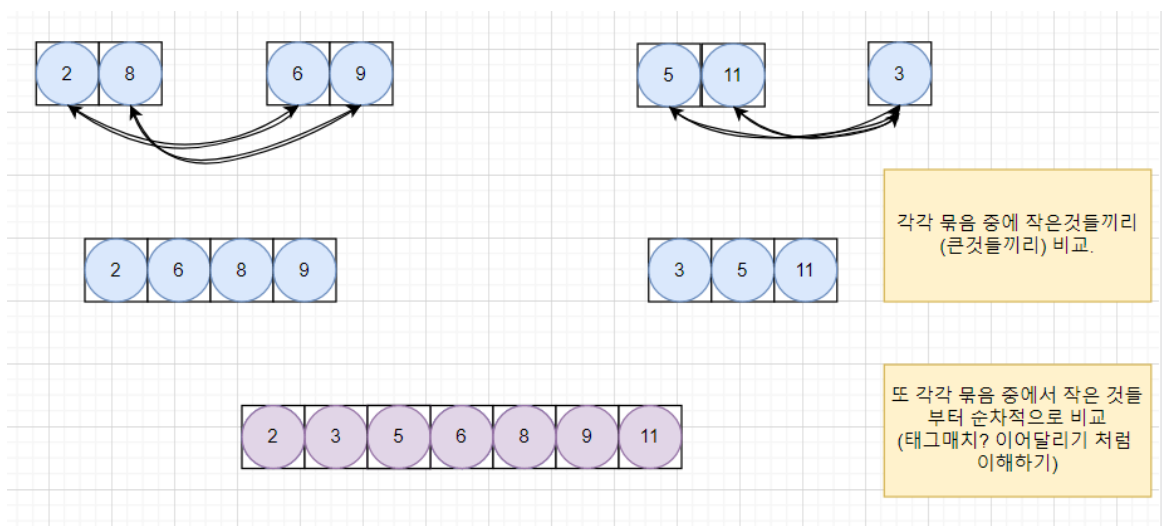
- 분할 정복 (divide and conquer)을 사용하는 방법이다. 분할 정복은 주어진 문제를 해결하기 쉬운 단계까지 분할 한 후에 분할 된 문제를 해결하고 그 결과를 다시 병합 하는 알고리즘이다.

### 1 - 데이터 분해



- 먼저 데이터들을 정렬을 생각하지 않고 1개씩 될때까지 나눈다.
- 나누는 방법은 배열을 반으로 쪼개고 그 쪼개진 배열을 또 다시 반으로 쪼개고 이 과정을 반복한다. **(나누기)**
- 위의 경우 7개의 데이터를 분할하기 위해 이 과정을 3번 거쳤다.

## 2 - 데이터를 비교하면서 결합.



- 분할이 완료 된 후에는 데이터를 비교하면서 결합한다.



- 제일 앞에 있는 2와 8을 합치는데, 작은 수 2가 앞으로, 큰 수 8이 뒤로 보낸 상태로 결합한다. 이 과정을 각각 2개씩 반복하여 합치고, 합친 2개를 다시 같은 방법으로 결합한다.

⇒ 이때, **각각의 그룹에서 먼저 제일 첫번째 값을 비교하고, 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.

## II. 데이터베이스

### 복습

- 자료형 : 많이 사용하는 자료형은 15개 전후.  
문자 데이터 : varchar  
긴 텍스트 : text있는데 varchar쓴다.  
숫자 : int  
날짜 : timestamp (메인) , datetime(날짜와 시간 자동입력)
- 테이블 작성 - 설계(브레인스토밍) → 정제 (테이블 설정을 구체화 하는것). → SQL 구문 생성.
- PRIMARY KEY → 자주 쓴다.
- 데이터 삽입 : **INSERT INTO 000**
- 테이블 조회 : SELECT 000  
⇒ 조건 달아서 조회도 가능.
- 주석 : 한줄 주석은 -- (빼기 2개), 긴주석은 /\* 000 \*/

- UPDATE → 게시물 업데이트
  - DELETE → 게시물 삭제. 회원 탈퇴.
  - DISTINCT → 중복제거
  - FROM → 테이블
  - WHERE → 원하지 않느걸 빼거나 원하는것만 보기 위해서.
- 중복은 AND, 혹은 OR
- ORDER BY - 생략하면 오름차순. 결과를 원하는 순서로 정렬.  
내림차순은 DESC.
  - 데이터베이스 중요하다.

## 1. 필터링 (Filtering)

- 새로운 데이터 웨어 하우스 피드를 준비할때 사용한 테이블에서 모든 데이터를 제거.
- 새 열이 추가 된 후, 테이블의 모든 행을 수정/제거 할 수 있다.
- 메세지 큐 테이블에서 모든 행 검색도 가능하다.

### • 조건 평가

! WHERE title = 'Teller' AND start\_date < '2007-01-01'

1. 타이들이 teller인 것.
2. start\_date가 2007년 1월 1일 미만인것.

⇒ 조건이 여러개 있어도 AND 연산자로 구분될 경우에는 결과셋에 모든 조건이 True인 경우만 포함된다. 즉, true로 평가된다.

- 괄호 사용.
  - 괄호 안의 조건이 먼저 평가 된다. 수학과 동일함.

Table 4-2. Three-condition evaluation using and, or

Intermediate result	Final result
WHERE true AND (true OR true)	True
WHERE true AND (true OR false)	True
WHERE true AND (false OR true)	True
WHERE true AND (false OR false)	False

⇒ 세가지 조건이 있을 경우 최종 결과는 괄호안의 두가지 조건의 결과의 최종 마지막 결과의 평가에 따라 최종 결과가 정해지게 된다.

- not 연산자 사용
  - not 연산자를 사용하여 평가결과를 반대로 한다.
  - true의 경우는 false로, false의 경우에는 true로 결과를 뒤집을 수 있다.
- 조건 작성
  - 표현식은 아래의 내용들로 구성될 수 있다.
    - 1) 숫자
    - 2) 테이블 또는 뷰의 컬럼 (뷰 : 테이블을 가공해서 또 하나의 테이블인것 처럼 쓰는 것.)
    - 3) 텔러와 같은 문자열
    - 4) concat과 같은 내장함수들.
    - 5) 서브쿼리 (지리 안의 지리), 헤드 텔러 등등과 같은 표현식 목록
- 조건의 유형
  - 1) 동등조건 : '열 = 값'. 어느 열에 가서 어느 값 찾아라.  
 ex) title = 'Teller', amount = 375.25  
 동등조건을 사용한 데이터 수정(삭제로 수정)
 

```
DELETE FROM account
WHERE status = 'CLOSED' AND YEAR(close_date) = 2002;
```
  - 2) 부등 조건 : 두 표현식이 동일하지 않을때 사용. <>

```
mysql> SELECT pt.name product_type, p.name product
-> FROM product p INNER JOIN product_type pt
-> ON p.product_type_cd = pt.product_type_cd
-> WHERE pt.name <> 'Customer Accounts';
```

product_type	product
Individual and Business Loans	auto loan
Individual and Business Loans	business line of credit
Individual and Business Loans	home mortgage
Individual and Business Loans	small business loan

4 rows in set (0.00 sec)

▼ 해설

product type을 p.name으로 지정.

p.name이 Customer Accounts가 아닌것을 고르기.

3) 범위 조건 : 특정 범위 내의 조건이 있는지를 확인하는 범위조건을 작성 할 수 있다.

이 유형은 보통 숫자 또는 시간 데이터로 작성할 때 주로 발생한다.

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

▼ 해설

출력은 emp\_id, fname, lname, start\_date

찾는 테이블은 employee.

start\_date가 2007-01-01보다 이전.

4) **between 연산자** : 범위의 상한과 하한이 모두 있을때 사용하여 하나의 조건으로 사용할 수 있다. (예제).

5) **wildcards** 사용 :

- ! 조건 ① 특정 문자로 시작 혹은 종료하는 문자열 (여러글자)
- ② 부분 문자열로 시작 혹은 종료 하는 문자열 (한글자)
  - ③ 문자열 내에 특정 문자를 포함하는 모든 문자열.
  - ④ 문자열 내에 특정 문자열을 포함하는 모든 문자열.
  - ⑤ 개별 문자에 관계 없이 특정한 형식의 문자열

⇒ (언더바)와 %를 쓴다.

(언더바) : 정확히 한글자

% : 0을 포함한 갯수에 상관 없이 모든 문자.

ex)

```
1 SELECT lname
2 FROM employee
3 WHERE lname LIKE '_a%e%';
```

employee (4r x 1c)	
lname	
Barker	
Hawthorne	
Parker	
Jameson	

#### ▼ 해설

empolyee 테이블에서 lname으로 정렬.

lname의 형태가 2번째 글자에는 a가 들어가고, 이후에 어디에든 e가 들어가 있는 것을 출력.

```
호스트: 127.0.0.1 | 데이터베이스: bank
```

```

1 SELECT cust_id, fed_id
2 FROM customer
3 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
6	666-66-6666
7	777-77-7777
8	888-88-8888
9	999-99-9999

▼ 해설

customer 테이블.

출력 열은 cust\_id, fed\_id

fed\_id의 값의 형태가 '(3자리)-(2자리)-(4자리)'의 형태인 것을 출력.

```
호스트: 127.0.0.1 | 데이터베이스: bank | 쿼리*
```

```

1 SELECT emp_id, fname, lname
2 FROM employee
3 WHERE lname LIKE 'F%' OR lname LIKE 'G%';

```

employee (4r × 3c)		
emp_id	fname	lname
5	John	Gooding
6	Helen	Fleming
9	Jane	Grossman
17	Beth	Fowler

▼ 해설

employee 테이블.

출력 열은 emp\_id, fname, lname.

lname의 첫 시작 문자가 F이거나 G인 열을 출력.

- null :

*Not applicable*

Such as the employee ID column for a transaction that took place at an ATM machine

*Value not yet known*

Such as when the federal ID is not known at the time a customer row is created

*Value undefined*

Such as when an account is created for a product that has not yet been added to the database

null의 3가지 뜻.

- ! 1) 해당사항이 없다.
- 2) 값을 아직 알 수 없다.
- 3) 값을 찾을 수 없다.

When working with null, you should remember:

- An expression can *be* null, but it can never *equal* null.
- Two nulls are never equal to each other.



⇒ null일 수는 있지만 null과 같은 값일 수는 없다.

⇒ 두개의 nulls는 서로 같지 않다. null은 서로 비교가 불가능 하다.

```
호스트: 127.0.0.1 | 데이터베이스: bank | 쿼리*  
1 SELECT emp_id, fname, lname, superior_emp_id  
2 FROM employee  
3 WHERE superior_emp_id IS NULL;
```

employee (1r × 4c)			
emp_id	fname	lname	superior_emp_id
1	Michael	Smith	(NULL)

문제 1.

ENAME	SAL	JOB
FORD	3000	ANALYST
SCOTT	3000	ANALYST



이름	월급
KING	5000
FORD	3000
SCOTT	3000

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

호스트: 127.0.0.1 데이터베이스

```

1 SELECT ENAME, SAL, JOB
2 FROM emp
3 WHERE sal = 3000;

```

emp (2r x 3c)

ENAME	SAL	JOB
FORD	3,000	ANALYST
SCOTT	3,000	ANALYST

문제 2

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

1	SELECT	ENAME,	SAL,	JOB,	HIREDATE,	DEPTNO
2	FROM	emp				
3	WHERE	ENAME	=	'SCOTT'		

emp (1r × 5c)					
ENAME	SAL	JOB	HIREDATE	DEPTNO	
SCOTT	3,000	ANALYST	1982-12-22	20	

```
SELECT ENAME, SAL, JOB, HIREDATE, DEPTNO
FROM emp
WHERE ENAME = 'SCOTT'
```

### 문제 3

연봉이 36000 이상인 직원들의 이름과 연봉을 출력해 보겠습니다.

ENAME	연봉
KING	60000
FORD	36000
SCOTT	36000

호스트: 127.0.0.1    데이터베이스: bank

```

1  SELECT ENAME, Sal*12 연봉
2  FROM emp
3  WHERE SAL*12 >= 36000;

```

emp (3r × 2c)

ENAME	연봉
KING	60,000
FORD	36,000
SCOTT	36,000

```

SELECT ENAME, Sal12 연봉
FROM emp
WHERE SAL12 >= 36000;

```

월급이 1000에서 3000 사이인 직원들의 이름과 월급을 출력해 보겠습니다.

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

```

1 SELECT ENAME, Sal
2 FROM emp
3 WHERE sal BETWEEN 1000 AND 3000;

```

emp (11r x 2c)

ENAME	Sal
BLAKE	2,850
CLARK	2,450
JONES	2,975
MARTIN	1,250
ALLEN	1,600
TURNER	1,500
WARD	1,250
FORD	3,000
SCOTT	3,000
ADAMS	1,100
MILLER	1,300

1982년도에 입사한 직원들의 이름과 입사일을 조회:

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

+ :: 1982년도에 입사한 직원들의 이름과 입사일을 조회:

```

1 SELECT ENAME, Hiredate
2 FROM emp
3 WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31';

```

emp (2r × 2c)

ENAME	Hiredate
SCOTT	1982-12-22
MILLER	1982-01-11

```

SELECT ENAME, Hiredate
FROM emp
WHERE hiredate BETWEEN '1982-01-01' AND '1982-12-31';

```

## Quiz 12

이름의 두 번째 철자가 M인 사원의 이름을 출력하시오

ENAME

SMITH

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

emp (1r x 1c)
ENAME
SMITH

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


이름의 두 번째 철자가 M인 사원의 이름을 출력해

ENAME
SMITH

이름이 A가 포함된 사원들을 전부 검색해

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%';
```



커미션이 NULL인 사원들의 이름과 커미션을 출력해 보겠습니다.

+ ::

ENAME	COMM
KING	
:	:
MILLER	

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

emp (10r x 2c)

ENAME	comm
KING	(NULL)
BLAKE	(NULL)
CLARK	(NULL)
JONES	(NULL)
JAMES	(NULL)
FORD	(NULL)
SMITH	(NULL)
SCOTT	(NULL)
ADAMS	(NULL)
MILLER	(NULL)

```
SELECT ENAME, comm
FROM emp
WHERE comm IS NULL;
```

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

```

1 SELECT ENAME, SAL, job
2 FROM emp
3 WHERE (JOB='SALESMAN' OR JOB='ANALYST' OR JOB='MANAGER');

```

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
SCOTT	3,000	ANALYST

```

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

```

호스트: 127.0.0.1	데이터베이스: bank	테이블: emp
----------------	--------------	----------

```

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

```

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
SCOTT	3,000	ANALYST

```

SELECT ENAME, SAL, job
FROM emp
WHERE JOB IN ('SALESMAN', 'ANALYST', 'MANAGER');

```

! IN 의 사용 방법

**SELECT \*FROM 테이블명**  
**WHERE 컬럼명IN (값1, 값2, ...);**

#### 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