MySQL Database

데이터 검색

SQL

Basic Query - SELECT 문의 기초

SELECT

- 데이터베이스에서 원하는 데이터를 검색, 추출
- Syntax

```
SELECT [ALL|DISTINCT] 열_리스트
     FROM  테이블_리스트
     [WHERE 조건]
     [GROUP BY 열_리스트 [HAVING 그룹 조건]]
     [ORDER BY 열_리스트 [ASC | DESC]];
```

- 기능
  - Projection : 원하는 컬럼 선택
  - Selection : 원하는 튜플 선택
  - Join : 두 개의 테이블 결합
  - 기타 : 각종 계산, 정렬, 요약(Aggregation)

SELECT의 기능

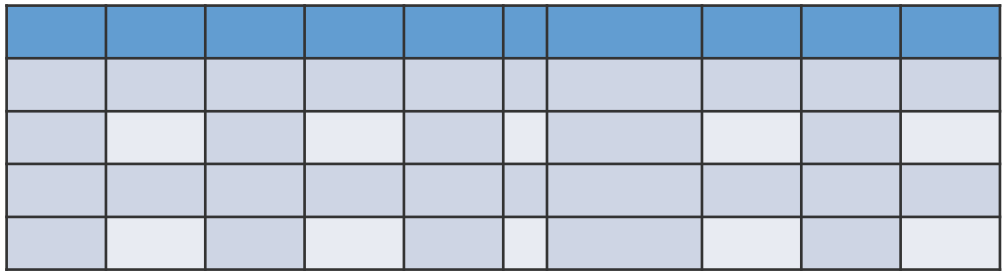
Projection

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	880		20
7499	ALLEN	SALESMAN	7698	81/02/20	1760	300	30
7521	WARD	SALESMAN	7698	81/02/22	1375	500	30
7566	JONES	MANAGER	7839	81/04/02	2975		20
7654	MARTIN	SALESMAN	7698	81/09/28	1375	1400	30
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7788	SCOTT	ANALYST	7566	87/04/19	3000		20
7839	KING	PRESIDENT		81/11/17	5000		10
7844	TURNER	SALESMAN	7698	81/09/08	1650	0	30
7876	ADAMS	CLERK	7788	87/05/23	1210		20
7900	JAMES	CLERK	7698	81/12/03	1045		30
7902	FORD	ANALYST	7566	81/12/03	3000		20
7934	MILLER	CLERK	7782	82/01/23	1430		10

Selection

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Join



기본 SELECT 문

- 형식

```
SELECT * | {[DISTINCT] column|expression [alias], …}
     FROM  table
```

- 내용 설명
  - * : 모든 컬럼 반환
  - DISTINCT : 중복된 결과 제거
  - SELECT 컬럼명 : Projection
  - FROM 대상 테이블
  - ALIAS : 컬럼 이름 변경(표시용)
  - Expression : 기본적인 연산 및 함수 사용 가능

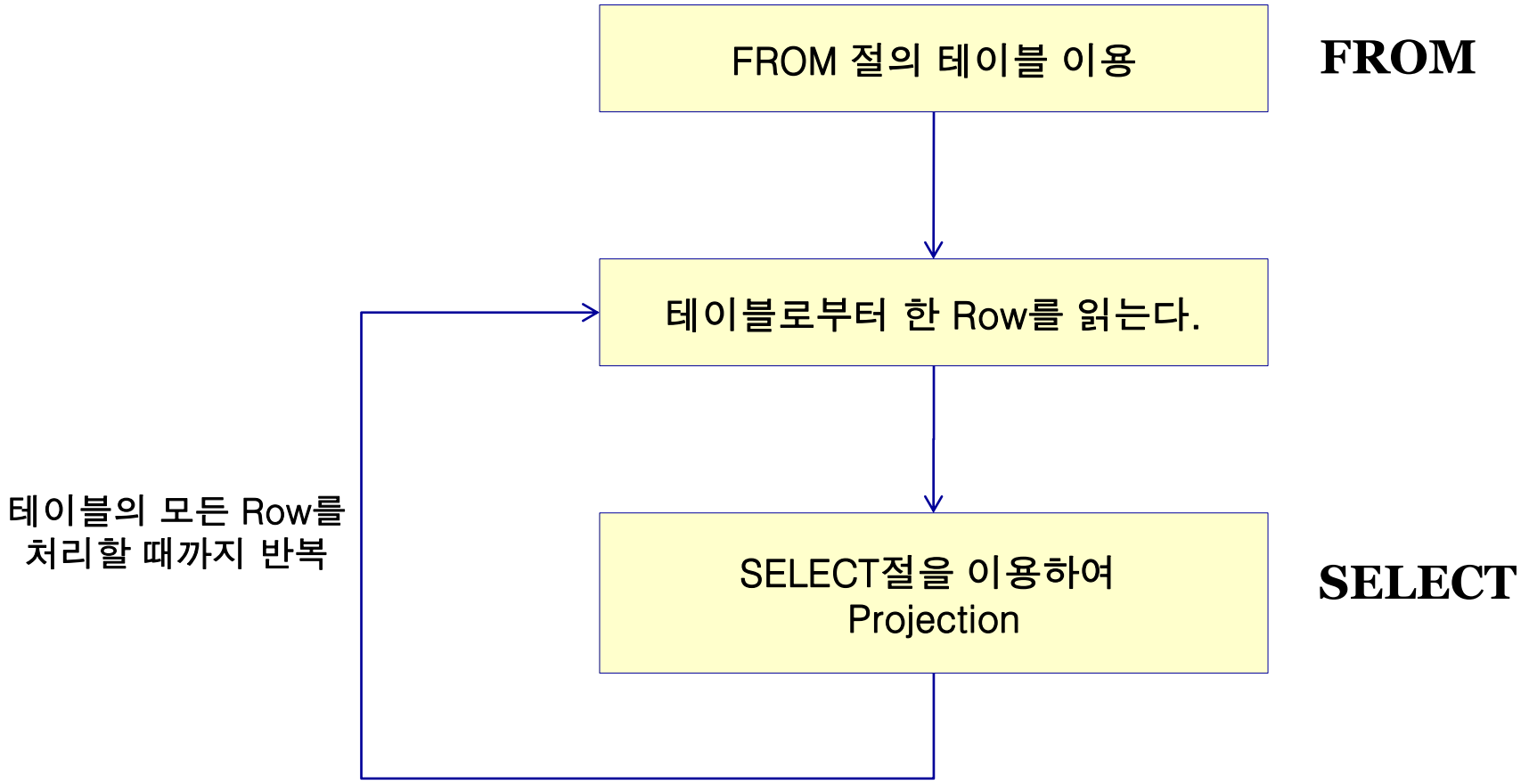
기본 SELECT 문의 예

- `SELECT * FROM employees;`
- `SELECT first_name FROM employees;`
- `SELECT first_name, gender FROM employees;`
- `SELECT first_name 이름 FROM employees;`

```
mysql> SELECT * FROM employees;
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
|  10002 | 1964-06-02 | Bezalel    | Simmel    | F      | 1985-11-21 |
|  10003 | 1959-12-03 | Parto      | Bamford   | M      | 1986-08-28 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |
|  10005 | 1955-01-21 | Kyoichi    | Maliniak  | M      | 1989-09-12 |
...
```

SELECT … FROM 절의 처리

: Flow



SELECT / FROM 절

- 모든 컬럼의 조회
  - 컬럼 목록에 *를 표시하면 테이블 내 모든 컬럼을 Projection 한다

```
SELECT * FROM {TABLE명};
```

  - SQL Recap:
    - 마지막은 세미콜론(;)
    - 대소문자 구분하지 않음

  - 테이블 employees의 모든 튜플을 불러와 모든 컬럼을 Projection 한다.

```
SELECT * FROM employees;
```

  - 테이블 departments의 모든 튜플을 불러와 모든 컬럼을 Projection 한다

```
SELECT * FROM departments;
```

SELECT / FROM 절

- 원하는 컬럼의 조회 (Projection을 원하는 컬럼명을 지정)
  - 컬럼 목록에 *를 표시하면 테이블 내 모든 컬럼을 Projection 한다

```
SELECT {컬럼명1}, {컬럼명2}, …
FROM {TABLE명};
```

  - 예: 다음 쿼리문을 보고 출력 결과를 예측해 봅시다

```
SELECT emp_no, first_name, last_name, hire_date
FROM employees;
```

SELECT / FROM 절

- SELECT ~ FROM 연습
  - 연습 1)
    - 사원의 이름(first_name)과 전화번호, 입사일, 급여를 출력해 봅시다
  - 연습 2)
    - 사원의 이름(first_name), 성(last_name), 급여, 전화번호, 입사일을 출력해 봅시다

산술연산(Arithmetic Operation)

- 기본적인 산술연산 사용 가능
  - +, -, *, /, 부호, 괄호 등
  - 우선순위 : 부호 -> 괄호 -> *, / -> +, -
  - 컬럼명, 숫자
  - 예
    - `SELECT salary / 12 FROM salaries;`

```
mysql> SELECT salary / 12 FROM salaries;
+--------------+
| salary / 12 |
+--------------+
|   5009.7500 |
|   5175.1667 |
|   5506.1667 |
|   5549.6667 |
|   5580.0833 |
...
```

SELECT / FROM 절
: 산술연산 연습

- 단순 수식 계산하기

```
SELECT {산술식} FROM dual;
```

```
mysql> SELECT 10 * 10 * 3.14159 FROM dual;
+--------------------+
| 10 * 10 * 3.14159 |
+--------------------+
|          314.15900 |
+--------------------+
```

  - dual : Pseudo Table. 특정 테이블이 아닌 MySQL 시스템으로부터 값을 가져올 때 사용 (생략 가능)

- 필드 값의 산술연산

```
SELECT salary / 12 FROM salaries;
SELECT salary + salary * 0.1 FROM salaries LIMIT 10;
```

NULL

- 아무런 값도 정해지지 않았음을 의미
- 어떠한 데이터타입에도 사용 가능
- NULL은 0이나 space 등과는 다르다
- NOT NULL 컬럼이나 Primary Key 속성의 컬럼에는 사용할 수 없음
- NULL을 포함한 산술식은 NULL

```
SELECT salary * NULL FROM employees;
```

- NULL인지 확인하려면 IS NULL, NULL이 아닌지 확인하려면 IS NOT NULL

Column Alias

- 컬럼의 출력 제목을 변경
- alias 내에 공백이다 특수문자를 포함하고자 한다면 큰따옴표(" ")를 사용
- 형태
  - `SELECT first_name name FROM employees;`
  - `SELECT first_name as name FROM employees;`
  - `SELECT first_name "name" FROM employees;`
  - `SELECT salary / 12 "Monthly Salary" FROM employees LIMIT 10;`

```
mysql> SELECT emp_no no, first_name as name FROM employees LIMIT 10;
+-------+-----------+
| no    | name      |
+-------+-----------+
| 10001 | Georgi    |
| 10002 | Bezalel   |
| 10003 | Parto     |
| 10004 | Chirstian |
| 10005 | Kyoichi   |
...
```

Literal

- SELECT 절에 사용되는 문자, 숫자, Date 타입 등의 상수
- Date 타입이나 문자열은 작은따옴표(' ')로 둘러싸야 함
- 문자열 결합(Concatenation) 함수 이용 : concat
- 예

```
mysql> SELECT concat( first_name, ' ', last_name ) AS 이름,
    ->  gender AS 성별,
    ->  hire_date AS 입사일
    -> FROM employees;
+--------------------------------+------+------------+
| 이름                           | 성별 | 입사일     |
+--------------------------------+------+------------+
| Georgi Facello                 | M    | 1986-06-26 |
| Bezalel Simmel                 | F    | 1985-11-21 |
| Parto Bamford                  | M    | 1986-08-28 |
| Chirstian Koblick              | M    | 1986-12-01 |
| Kyoichi Maliniak               | M    | 1989-09-12 |
```

DISTINCT
: 중복행의 제거

- 중복되는 행이 출력되는 경우, DISTINCT 키워드로 중복행을 제거할 수 있음
  - 예제 1: employees 테이블에서 모든 job_id 출력

```
SELECT job_id FROM employees ;
```

  - 예제 2: employees 테이블에서 job_id은 어떤 것이 있는지 job_id를 중복 없이 한 번씩만 출력

```
SELECT DISTINCT job_id FROM employees;
```

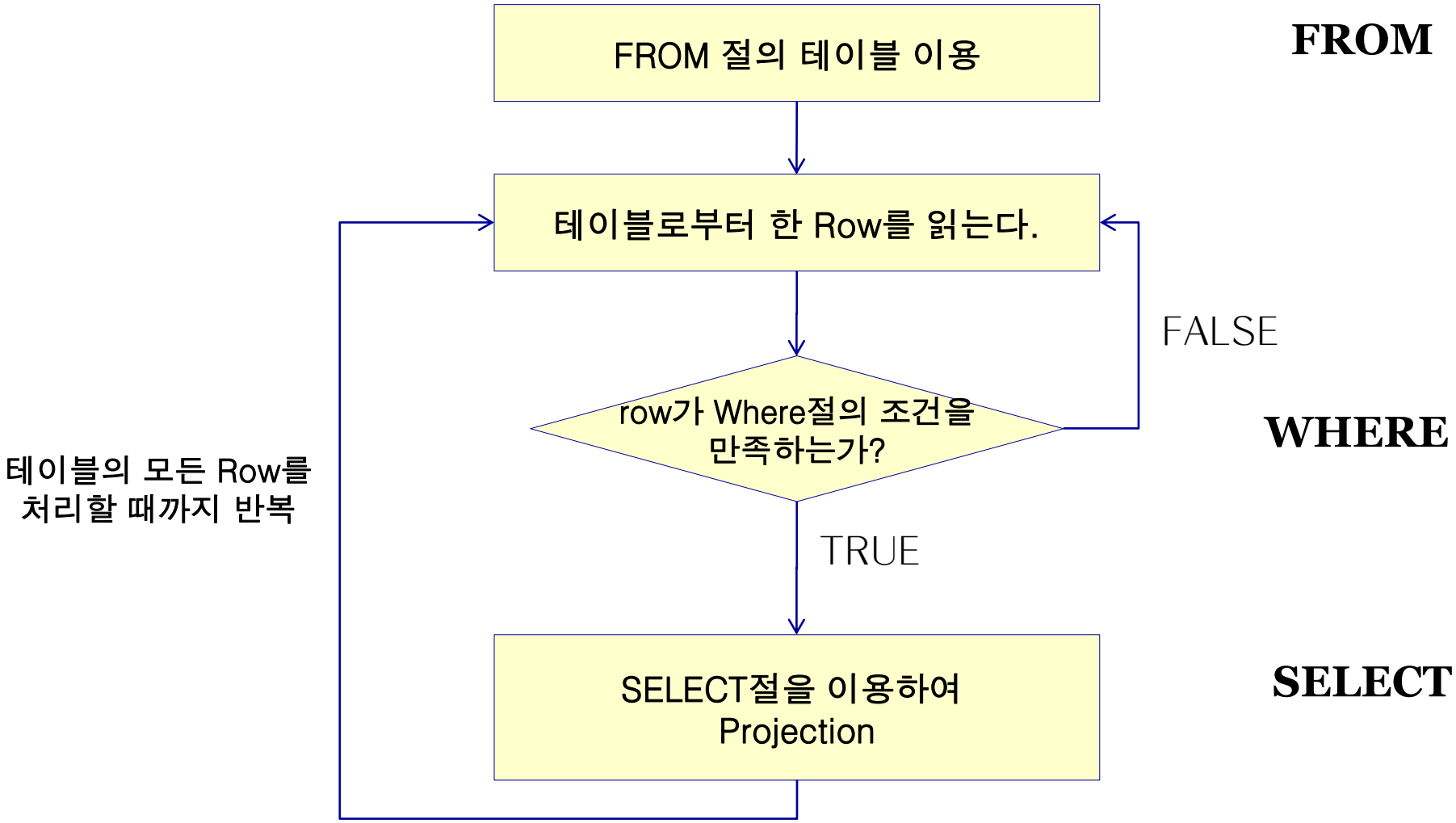
  - 두 쿼리를 각각 입력해보고 출력 결과에 대해 생각해 봅시다

WHERE

- 조건을 부여하여 만족하는 ROW를 선택(Selection)
- 연산자
  - =, !=, >, <, <=, >=
  - IN : 집합에 포함되는가?
  - BETWEEN a AND b : a와 b 사이인가?
  - IS NULL, IS NOT NULL : NULL 여부 검사
  - AND, OR : 둘 다 만족? 둘 중 하나만 만족?
  - NOT : 만족하지 않음?
  - ANY, ALL : 집합 중 어느 한 열, 집합 중 모든 열 (다른 비교 연산자와 함께 사용)
  - EXIST : 결과 Row가 한 개 이상 있는가? (Subquery에서 사용)

WHERE 절의 처리

: Flow



WHERE 절
: 비교 연산자

- 산술 비교 연산자

```
SELECT department_name
FROM departments
WHERE department_id = 10;
```

```
SELECT first_name, salary
FROM employees
WHERE salary > 150000;
```

Operator	Purpose
=	Equal to
<>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

```
SELECT concat(first_name, ' ', last_name) AS full_name,
       job_id AS  "직군명",
       hire_date AS "입사일"
FROM employees
WHERE hire_date < '1991-01-01';
```

WHERE 절
: 비교 연산자

- 산술 비교 연산자 IN
  - 제공되는 리스트 항목에 해당하는 값이 있으면 반환

```
SELECT employee_id, department_id
FROM employees
WHERE department_id IN (10, 20, 30);
```

  - <-> NOT IN

```
SELECT employee_id, department_id
FROM employees
WHERE department_id NOT IN (10, 20, 30);
```

WHERE 절
: 비교 연산자

- 산술 비교 연산자 BETWEEN ~ AND ~
  - 주어지는 두 값 사이에 해당하는 값이면 반환

```
SELECT first_name, salary
FROM employees
WHERE salary BETWEEN 14000
       AND 17000;
```

LIKE 연산

- Wildcard를 이용한 문자열 부분 매칭
- Wildcard
  - % : 임의의 길이의 문자열(공백 문자 포함)
  - _ : 한 글자
- Escape
  - ESCAPE 뒤의 문자열로 시작하는 문자는 Wildcard 가 아닌 것으로 해석
- 예
  - first_name LIKE 'KOR%' : KOR로 시작하는 모든 문자열 (KOR 포함)
  - first_name LIKE 'KOR_' : KOR 뒤에 하나의 문자가 오는 모든 문자열
  - first_name LIKE 'KOR/%%' ESCAPE '/' : 'KOR%'로 시작하는 모든 문자열

논리 연산자

- 논리 연산자를 이용하여 복잡한 질의 조건을 줄 수 있으며, 전체 조건식의 연산 결과가 TRUE인 Row만 선택된다

1	Arithmetic operators
2	Comparison operators
3	NOT
4	AND
5	OR

논리 연산자의 결과값

- NULL을 주의
  - NULL이 있으면 기본적으로 NULL, 확실히 답이 나오는 경우만 계산 가능

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

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

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

WHERE 절
: 논리 연산자 연습

▶ 조건이 두 개 이상일 때 한꺼번에 조회하기

```
SELECT emp_no, salary
FROM salaries
WHERE salary >= 150000
      AND salary <= 160000;
```

- 급여가 150000 이상 160000이하인 사원의 급여를 출력

WHERE 절
: BETWEEN 연습

- BETWEEN 연산자를 이용한 특정 구간의 값 출력

```
SELECT emp_no, salary
FROM salaries
WHERE salary BETWEEN 150000
                 AND 160000;
```

- 급여가 150000 이상 160000이하인 사원의 급여를 출력

- 앞서 비교 연산자를 이용한 쿼리문과 비교해 봅시다
- 작은 값을 앞쪽에, 큰 값을 뒤쪽에 부여
- 유용하지만 느린 연산자에 속함

WHERE 절
: IN 연습

- 여러 조건을 검사하기 위해 IN 연산자를 활용합니다

```
SELECT first_name, last_name, hire_date
FROM employees
WHERE first_name IN ('Steve', 'Adam');
```

  - 동일 내용의 쿼리를 단순 비교 연산자로 작성해 보고 그 차이점을 비교해 봅시다

WHERE 절
: Like 연습

- SQL Recap:
  - % : 임의의 길이의 문자열(공백 문자 가능)
  - _ : 임의의 한 글자

- [연습] 이름에 am을 포함하고 있는 사원의 이름과 연봉을 출력
- [연습] 이름의 두 번째 글자가 a인 사원의 이름과 연봉을 출력

연산자 우선 순위

1. Arithmetic Operators
2. Concatenation Operators
3. Comparison Conditions
4. IS [NOT] NULL, LIKE, [NOT] IN
5. [NOT] BETWEEN
6. NOT Logical condition
7. AND Logical condition
8. OR logical condition

ORDER BY

- 주어진 컬럼 리스트의 순서로 결과를 정렬
- 결과 정렬 방법
  - ASC : 오름차순 (작은값 -> 큰값) - default
  - DESC : 내림차순 (큰 값 -> 작은 값)
- 정렬 기준은 여러 컬럼에 지정 가능
  - 예) SELECT * FROM employees ORDER BY first_name, hire_date DESC LIMIT 10;
    : 의미 해석 = 이름 순으로 정렬한 후, hire_date가 최신인 사람부터 출력
  - MySQL Tip: 결과를 무작위 순으로 배열하고자 할 때, ORDER BY RAND() 를 사용

ORDER BY

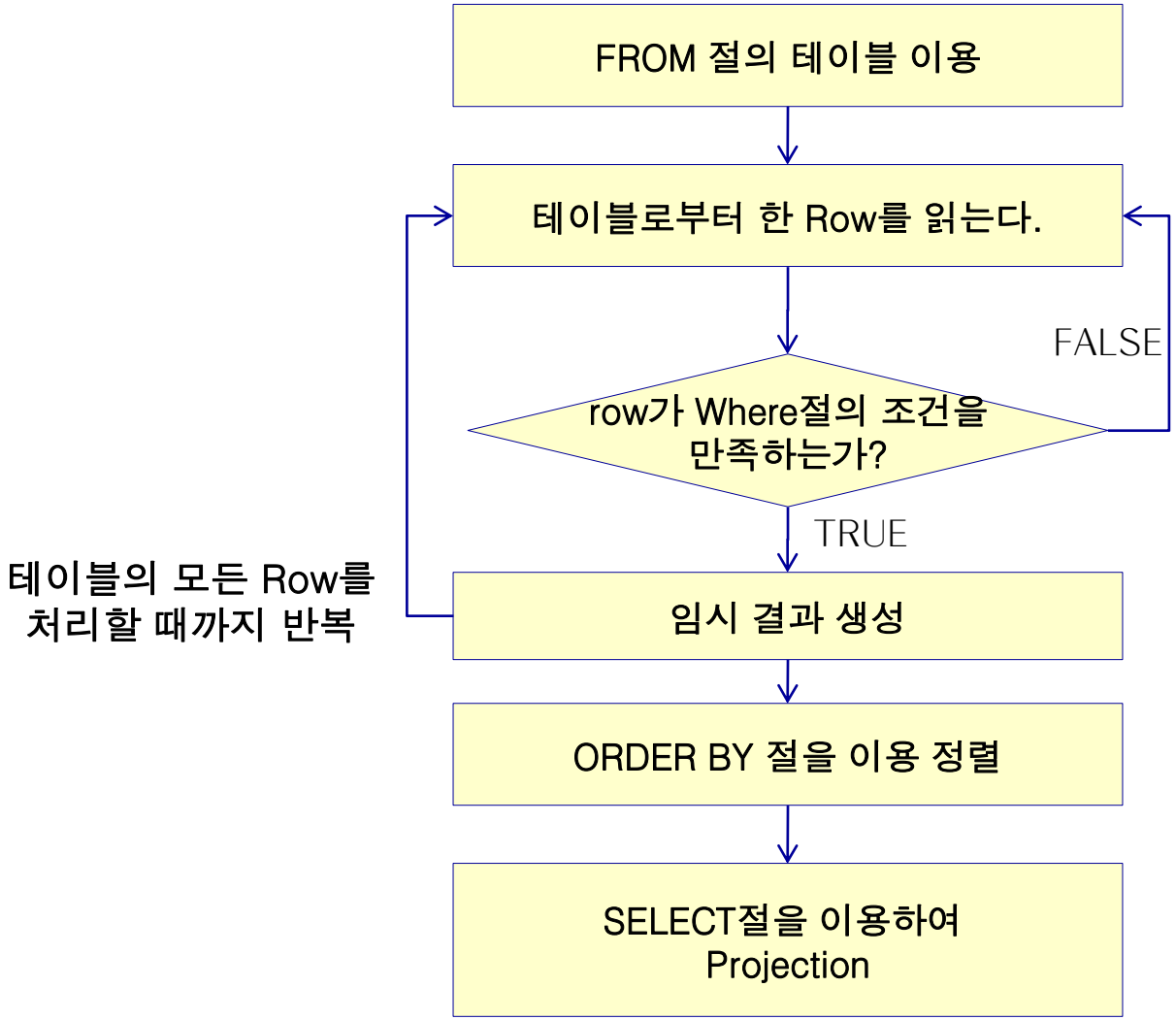
- ORDER BY 절을 이용하여 데이터를 사용하기 편리하게 정렬하기

```
SELECT concat(first_name, ' ', last_name) AS 이름,
       gender AS 성별,
       hire_date AS 입사일
FROM employees
ORDER BY hire_date;
```

- 기본값: 오름차순 (ASC: 작은 값 -> 큰 값 순) <-> 내림차순(DESC)은 반대
  - 한글: 가, 나, 다, 라 …
  - 영어: A, B, C, D
  - 숫자: 1, 2, 3, 4
  - 날짜 : 오래된 날짜 -> 최근 날짜 순

  - 정렬 조건이 복수일 경우는 ,로 구분하여 명시

ORDER BY 절의 처리



Functions

단일행 함수 (Single Row Function)

SQL Functions

- Single-Row Function : 하나의 Row를 입력으로 받는 함수
  - 숫자 함수
  - 문자 함수
  - 날짜 함수
  - 변환 함수
  - 기타 함수
- Aggregation Function : 집합 함수
- Analytic Function : 분석 함수
- Regular Expression : 정규표현식 (Oracle 10g 이상)

단일행 함수
: 문자열 함수

- UCASE(s), UPPER(s) : 문자열 s를 대문자로 변환

```
mysql> SELECT UPPER("SEoul"), UCASE("seOUL");
+----------------+----------------+
| UPPER("SEoul") | UCASE("seOUL") |
+----------------+----------------+
| SEOUL          | SEOUL          |
+----------------+----------------+
1 row in set (0.00 sec)
```

- LCASE(s), LOWER(s) : 문자열 s를 소문자로 변환

```
mysql> SELECT LOWER("SEoul"), LCASE("seOUL");
+----------------+----------------+
| LOWER("SEoul") | LCASE("seOUL") |
+----------------+----------------+
| seoul          | seoul          |
+----------------+----------------+
1 row in set (0.00 sec)
```

단일행 함수
: 문자열 함수

- SUBSTRING(s, m, n) : 문자열 일부를 추출 m번째 문자부터 n개

```
mysql> SELECT SUBSTRING("Happy Day", 3, 2);
+-------------------------------+
| SUBSTRING("Happy Day", 3, 2) |
+-------------------------------+
| pp                            |
+-------------------------------+
1 row in set (0.00 sec)
```

- 예제: employees 테이블에서 1989년 입사한 직원의 이름, 입사일을 출력

```
SELECT concat(first_name, ' ', last_name) AS name,
       hire_date
FROM employees
WHERE SUBSTRING(hire_date, 1, 4) = "1989";
```

단일행 함수
: 문자열 함수

- REPLACE(s, p, r) : 문자열 s 일부를 다른 문자열로 치환 - p 문자열을 r 문자열로 대치

```
mysql> SELECT REPLACE("Happy Day", "Happy", "Sad");
+---------------------------------------+
| REPLACE("Happy Day", "Happy", "Sad") |
+---------------------------------------+
| Sad Day                               |
+---------------------------------------+
1 row in set (0.00 sec)
```

단일행 함수
: 문자열 함수

- LPAD(s1, n, s2), RPAD(s1, n, s2) : 문자열 s1의 좌, 우 빈 공간을 지정한 문자 s2로 채움
  - n은 총 출력 문자열 자릿수

```
mysql> SELECT LPAD('hi', 5, '?'), LPAD('joe', 7, '*');
+--------------------+----------------------+
| LPAD('hi', 5, '?') | LPAD('joe', 7, '*') |
+--------------------+----------------------+
| ???hi              | ****joe              |
+--------------------+----------------------+
1 row in set (0.00 sec)
```

- 예제: 다음 쿼리를 보고 출력 결과를 예측해 봅시다

```
mysql> SELECT RPAD('hi', 5, '?'), RPAD('joe', 7, '*');
```

단일행 함수
: 문자열 함수

- LPAD, RPAD
  - 예제: salaries 테이블에서 2001년 급여가 70000불 이하인 직원만 사번, 급여를 출력하되 급여는 10자리로 부족한 자릿수는 *로 출력하십시오.

```
mysql> SELECT emp_no, LPAD(salary, 10, '*')
    -> FROM salaries
    -> WHERE from_date like '2001-%'
    ->       AND salary =< 70000;
+--------+-----------------------+
| emp_no | LPAD(salary, 10, '*') |
+--------+-----------------------+
|  10003 | *****43311            |
|  10006 | *****59755            |
|  10012 | *****54423            |
|  10013 | *****68901            |
|  10014 | *****60598            |
...
```

단일행 함수
: 문자열 함수

- TRIM(s), LTRIM(s), RTRIM(s) : 문자열 좌우의 지정한 문자열 제거(기본값: 공백문자)

```
mysql> SELECT LTRIM(' hello '), RTRIM(' hello ');
+------------------+------------------+
| LTRIM(' hello ') | RTRIM(' hello ') |
+------------------+------------------+
| hello            |  hello           |
+------------------+------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT TRIM(' hello '), TRIM(BOTH 'x' FROM 'xxxxhelloxxxx');
+-----------------+-------------------------------------+
| TRIM(' hello ') | TRIM(BOTH 'x' FROM 'xxxxhelloxxxx') |
+-----------------+-------------------------------------+
| hello           | hello                               |
+-----------------+-------------------------------------+
1 row in set (0.00 sec)
```

BOTH 대신, LEADING, TRAILING으로 바꾸어 각각 테스트 해 봅시다

단일행 함수
: 문자열 함수

- String Functions ReCap:
  - https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

Function	설명
CONCAT(s1,s2)	문자열 결함
LOWER(s), LCASE(s)	소문자로 변경
UPPER(s), UCASE(s)	대문자로 변경
LPAD(s1,n,s2)	문자열의 왼쪽 채움 (길이:n, 채움문자 s2)
RPAD(s1,n,s2)	문자열 오른쪽 채움 (길이:n, 채움문자 s2)
LTRIM(s,c)	문자열 왼쪽 c문자열 제거
RTRIM(s,c)	문자열 오른쪽 c문자열 제거
REPLACE(s,p,r)	문자열 치환, S속의 p문자열을 r로 치환
SUBSTRING(s,m,n)	부분 문자열, m번째부터 길이 n인 문자열 반환
LENGTH(s)	문자열 길이 반환

단일행 함수
: 수치형 함수

- ABS(x) : x의 절대값을 구함

```
mysql> SELECT ABS(2), ABS(-2);
+--------+---------+
| ABS(2) | ABS(-2) |
+--------+---------+
|      2 |       2 |
+--------+---------+
1 row in set (0.00 sec)
```

- MOD(n, m) : n을 m으로 나눈 나머지 값을 반환 = %

```
mysql> SELECT MOD(234, 10), 253 % 7, MOD(29, 9);
+--------------+---------+------------+
| MOD(234, 10) | 253 % 7 | MOD(29, 9) |
+--------------+---------+------------+
|            4 |       1 |          2 |
+--------------+---------+------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- FLOOR(x) : x 보다 크지 않은 가장 큰 정수를 반환. BIGINT로 자동 변환됨

```
mysql> SELECT FLOOR(1.23), FLOOR(-1.23);
+-------------+--------------+
| FLOOR(1.23) | FLOOR(-1.23) |
+-------------+--------------+
|           1 |           -2 |
+-------------+--------------+
1 row in set (0.00 sec)
```

- CEILING(x) : x 보다 작지 않은 가장 작은 정수를 반환

```
mymysql> SELECT CEILING(1.23), CEILING(-1.23);
+---------------+----------------+
| CEILING(1.23) | CEILING(-1.23) |
+---------------+----------------+
|             2 |             -1 |
+---------------+----------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- ROUND(x) : x 에 가장 근접한 정수를 반환 (반올림)

```
mysql> SELECT ROUND(-1.23), ROUND(-1.58), ROUND(1.58);
+--------------+--------------+-------------+
| ROUND(-1.23) | ROUND(-1.58) | ROUND(1.58) |
+--------------+--------------+-------------+
|           -1 |           -2 |           2 |
+--------------+--------------+-------------+
1 row in set (0.00 sec)
```

- ROUND(x, d) : x 값 중에서 소수점 d 자리에 가장 근접한 수를 반환 (소수점 반올림)

```
mysql> SELECT ROUND(1.298, 1), ROUND(1.298, 0);
+-----------------+-----------------+
| ROUND(1.298, 1) | ROUND(1.298, 0) |
+-----------------+-----------------+
|             1.3 |               1 |
+-----------------+-----------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- TRUNCATE(x, d) : x의 소수점 d 자리 아래 값을 버림

```
mysql> SELECT ROUND(1.298, 1), TRUNCATE(1.298, 1);
+------------------+--------------------+
| ROUND(1.298, 1) | TRUNCATE(1.298, 1) |
+------------------+--------------------+
|              1.3 |                1.2 |
+------------------+--------------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- SIGN(x) : x 가 음수이면 -1, 0이면 0, 양수이면 1을 반환

```
mysql> SELECT SIGN(-32), SIGN(0), SIGN(32);
+-----------+---------+----------+
| SIGN(-32) | SIGN(0) | SIGN(32) |
+-----------+---------+----------+
|        -1 |       0 |        1 |
+-----------+---------+----------+
1 row in set (0.00 sec)
```

- POW(x, y), POWER(x, y) : x의 y 제곱을 반환

```
mysql> SELECT POW(2, 2), POWER(2, -2);
+-----------+--------------+
| POW(2, 2) | POWER(2, -2) |
+-----------+--------------+
|         4 |         0.25 |
+-----------+--------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- GREATEST(x, y, ...) : 주어진 값 중에서 가장 큰 값을 반환

```
mysql> SELECT GREATEST(2, 1, 0), GREATEST(4.0, 5.0, 3.0), GREATEST('B', 'A', 'C');
+------------------+--------------------------+-------------------------+
| GREATEST(2, 1, 0) | GREATEST(4.0, 5.0, 3.0) | GREATEST('B', 'A', 'C') |
+------------------+--------------------------+-------------------------+
|                 2 |                      5.0 | C                       |
+------------------+--------------------------+-------------------------+
1 row in set (0.00 sec)
```

- LEAST(x, y, ...) : 주어진 값 중에서 가장 작은 값을 반환

```
mysql> SELECT LEAST(2, 1, 0), LEAST(4.0, 5.0, 3.0), LEAST('B', 'A', 'C');
+---------------+-----------------------+----------------------+
| LEAST(2, 1, 0) | LEAST(4.0, 5.0, 3.0) | LEAST('B', 'A', 'C') |
+---------------+-----------------------+----------------------+
|              0 |                   3.0 | A                    |
+---------------+-----------------------+----------------------+
1 row in set (0.00 sec)
```

단일행 함수
: 수치형 함수

- Numeric Functions ReCap:
  - https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html

Function	설명	Example	Result
ABS(n)	절대값	ABS(-5)	5
CEILING(n)	n 보다 크거나 같은 최소 정수	CEIL(-2.4)	-2
FLOOR(n)	n 보다 작거나 같은 최대 정수	FLOOR(-2.4)	-3
MOD(m,n)	나머지	MOD(13,2)	1
POWER(m,n)	m의 n승	POWER(2,3)	8
ROUND(m,n)	소수점아래 n자리까지 반올림	ROUND(4.567,2)	4.57
TRUNCATE(m,n)	소수점아래 n자리미만 버림	TRUNC(4.567,2)	4.56
SIGN(n)	부호 (1, 0, -1)	SIGN(-10)	-1

단일행 함수
: 날짜형 함수

- CURDATE(), CURRENT_DATE : 오늘 날짜를 YYYY-MM-DD나 YYYYMMDD 형식으로 반환

```
mysql> SELECT CURDATE(), CURRENT_DATE;
+------------+--------------+
| CURDATE()  | CURRENT_DATE |
+------------+--------------+
| 2018-10-21 | 2018-10-21   |
+------------+--------------+
1 row in set (0.00 sec)
```

- CURTIME(), CURRENT_TIME : 현재 시간을 HH:MM:SS나 HHMMSS 형식으로 반환

```
mysql> SELECT CURTIME(), CURRENT_TIME;
+-----------+--------------+
| CURTIME() | CURRENT_TIME |
+-----------+--------------+
| 17:17:09  | 17:17:09     |
+-----------+--------------+
1 row in set (0.00 sec)
```

단일행 함수
: 날짜형 함수

- NOW(),
  SYSDATE(),
  CURRENT_DATE : 오늘 현재 시간을 YYYY-MM-DD HH:MM:SS나
  YYYYMMDDHHMMSS 형식으로 반환

```
mysql> SELECT NOW(), SYSDATE(), CURRENT_TIMESTAMP;
+---------------------+---------------------+---------------------+
| NOW()               | SYSDATE()           | CURRENT_TIMESTAMP   |
+---------------------+---------------------+---------------------+
| 2018-10-21 17:19:58 | 2018-10-21 17:19:58 | 2018-10-21 17:19:58 |
+---------------------+---------------------+---------------------+
1 row in set (0.00 sec)
```

단일행 함수
: 날짜형 함수

- DATE_FORMAT(date, format) : 입력된 date를 format 형식으로 변환

```
mysql> SELECT DATE_FORMAT(CURDATE(), "%W %M %Y");
+--------------------------------------+
| DATE_FORMAT(CURDATE(), "%W %M %Y") |
+--------------------------------------+
| Sunday October 2018                  |
+--------------------------------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT DATE_FORMAT(CURDATE(), "%Y.%m.%d");
+--------------------------------------+
| DATE_FORMAT(CURDATE(), "%Y.%m.%d") |
+--------------------------------------+
| 2018.10.21                           |
+--------------------------------------+
1 row in set (0.00 sec)
```

단일행 함수
: 날짜형 함수

- PERIOD_DIFF(p1, p2) : YYMM이나 YYYYMM으로 표기되는 p1과 p2의 차이 개월을 반환

- 예제: 각 직원들에 대해 직원 이름과 근무 개월 수를 출력

```
SELECT CONCAT(first_name, ' ', last_name) AS name,
       PERIOD_DIFF(DATE_FORMAT(CURDATE(), '%Y%m'),
                   DATE_FORMAT(hire_date, '%Y%m'))
FROM employees;
```

단일행 함수
: 날짜형 함수

- DATE_ADD(date, INTERVAL expr type)
  DATE_SUB(date, INTERVAL expr type)
  ADDDATE(date, INTERVAL expr type)
  SUBDATE(date, INTERVAL expr type)
  - 날짜 date에 type 형식으로 지정한 expr 값을 더하거나 뺀다
  - DATE_ADD()와 ADDDATE()는 같은 의미
    DATE_SUB()와 SUBDATE()는 같은 의미

- 예제: 각 직원들의 이름과 입사일, 입사 후 1개월 후 날짜를 출력

```
SELECT first_name, hire_date, DATE_ADD(hire_date, INTERVAL 1 MONTH)
FROM employees;
```

- 날짜형 함수 : https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

단일행 함수
: 변환함수 - CAST

- CAST 함수 : type을 변경(지정)하는데 활용하는 함수
- CAST 함수의 사용법

```
CAST(expression AS type)
CONVERT(expression, type)
```

- MySQL의 주요 Type
  - BINARY
  - CHAR
  - DATE
  - DATETIME
  - TIME
  - SIGNED {INTEGER}
  - UNSIGNED {INTEGER}

단일행 함수
: 변환함수 - CAST

- 예제

```
mysql> SELECT CAST(now() AS DATE);
+----------------------+
| CAST(now() AS DATE) |
+----------------------+
| 2018-10-22           |
+----------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CAST("123" AS UNSIGNED);
+--------------------------+
| CAST("123" AS UNSIGNED) |
+--------------------------+
|                      123 |
+--------------------------+
1 row in set (0.00 sec)
```