MySQL & Maria DB

SQL 기본

목차

- 기본적인 데이터베이스 쿼리
- 데이터 형
- 데이터를 조작하기 위한 **SELECT** 문
- **DISTINCT** 키워드

기본적인 데이터베이스 쿼리

- 스키마 정보 알아보기
  - 다음 쿼리를 통해 내부 스키마의 정보에 대해 알 수가 있습니다.

형식	SHOW DATABASES;

  - show databases 명령어는 현재 내부에 있는 스키마 목록을 출력합니다.
  - 원하는 스키마를 사용하기 위해 **use**라는 명령어 사용이 가능합니다.

형식	USE [스키마 명];

기본적인 데이터베이스 쿼리

- 테이블 정보 알아보기
  - 아래 쿼리를 통해 현재 스키마에 있는 테이블 정보들을 볼 수 있습니다.

형식	SHOW tables;

  - 테이블에서 데이터를 조회하기 위해서는 테이블의 구조를 알아야 합니다.
  - 테이블의 구조를 확인하기 위한 명령어로는 DESCRIBE가 있습니다.

형식	DESC [테이블명];

  - DESC 명령어는 테이블의 컬럼 이름, 데이터 형, 길이와 NULL 허용 유무 등과 같은 특정 테이블의 정보를 알려줍니다.

기본적인 데이터베이스 쿼리

- 테이블 정보 알아보기
  - 학습용으로 제공되는 DEPT 테이블은 부서의 정보를 저장하고 있으며, 이에 대한 구조를 살펴보기 위해서는 desc 명령어를 사용해야 합니다.

```
desc dept;
```

	Field	Type	Null	Key	Default	Extra
1	DEPTNO	int(11)	NO	PRI	<null>	
2	DNAME	varchar(14)	YES		<null>	
3	LOC	varchar(13)	YES		<null>	

  - DESC 명령어는 테이블의 컬럼 이름, 데이터 형, 길이와 NULL 허용 유무 등과 같은 특정 테이블의 정보를 알려줍니다.

기본적인 데이터베이스 쿼리

- 테이블 정보 알아보기
  - DESC 명령어로 DEPT 테이블을 살펴보면 3개의 컬럼으로 구성되어 있음을 살펴볼 수 있습니다.

칼럼 이름	의미
DEPTNO	부서번호
DNAME	부서명
LOC	지역명

기본적인 데이터베이스 쿼리

- 테이블 정보 알아보기

예	DESC EMP ---- ***EMP테이블의 구조 살피기***

	Field	Type	Null	Key	Default	Extra
1	EMPNO	int(11)	NO	PRI	<null>	
2	ENAME	varchar(10)	YES		<null>	
3	JOB	varchar(9)	YES		<null>	
4	MGR	int(11)	YES		<null>	
5	HIREDATE	datetime	YES		<null>	
6	SAL	double	YES		<null>	
7	COMM	double	YES		<null>	
8	DEPTNO	int(11)	YES	MUL	<null>	

칼럼 이름	의미	칼럼 이름	의미
EMPNO	사원번호	HIREDATE	입사일
ENAME	사원이름	SAL	급여
JOB	담당 업무	COMM	커미션
MGR	해당 사원의 상사 사원 번호	DEPTNO	부서번호

데이터 형

- 숫자 데이터 형식
  - DECIMAL 형식은 정확한 수치를 저장하고 FLOAT, REAL 형식은 근사치를 저장합니다
  - 소수점이 있는 실수는 되도록 DECIMAL 형식을 사용하여 저장하는 것이 바람직합니다
  - 예를 들어 -999999.99 ~ 999999.99 범위의 숫자를 저장할 때는 DECIAML(9,2)로 설정합니다
  - 어떤 숫자를 부호 없는 정수로 지정하면
    - TINYINT는 0~255, SMALLINT는 0~65535,
    - MEDIUMINT는 0~16777235,
    - INT는 0~약 42억
    - BIGINT는 0~약 1800경으로 표현할 수 있음
  - 부모 없는 정수를 저장할 때는 뒤에 UNSIGNED 예약어를 붙입니다.

데이터 형

- 숫자 데이터 형식

데이터 형식	바이트 수	숫자 범위	설명
BIT(N)	N/8		• 1~64bit 표현 • b'0000' 형식으로 저장
TINYINT	1	–128~127	• 정수 저장
BOOL BOOLEAN	1	–128~127	• 정수 저장 • TINYINT(1)과 동일 • 0은 false로, 그 외는 true로 취급
SMALLINT	2	–32768~32767	• 정수 저장
MEDIUMINT	3	–8388608~8388607	• 정수 저장
INT INTEGER	4	약 –21억~21억	• 정수 저장
BIGINT	8	약 –900경~900경	• 정수 저장
FLOAT	4	–3.40E+38~–1.17E–38	• 소수점 이하 7자리까지 저장
DOUBLE REAL	8	–1.22E–308~1.79E+308	• 소수점 이하 15자리까지 저장
DECIMAL(m,[d]) DEC(m,[d]) FIXED(m,[d]) NUMERIC(m,[d])	5~17	–1038+1~1038–1	• 전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자 저장 예: DECIMAL(5,2)는 전체 자릿수를 5자리로 하되, 그중 소수점 이하를 2자리로 하겠다는 뜻

데이터 형

- 문자 데이터 형식
  - CHAR 형식은 고정 길이 문자열을 저장하고 자릿수가 고정되어 있습니다.
  - VARCHAR 형식은 가변 길이 문자형을 저장합니다.
  - BINARY와 VARBINARY 형식은 바이트 형식의 이진 데이터 값을 저장합니다.
  - TEXT 형식은 대용량 글자를 저장하기 위한 형식으로, 필요한 크기에 따라서 TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT 등의 형식을 사용할 수 있습니다.
  - BLOB(Binary Large OBject) 형식은 사진, 동영상, 문서 파일 등의 대용량 이진 데이터를 저장할 때 사용합니다.
  - ENUM 형식은 열거형 데이터를 저장하는데 사용합니다.
  - SET 형식은 최대 64개의 데이터를 2개씩 세트로 묶어서 저장할 때 사용합니다.

데이터 형

- 문자 데이터 형식

데이터 형식		바이트 수	설명
CHAR(n)		1~255	• 고정 길이 문자형 저장(character의 약자) • n을 1~255까지 지정 • CHAR만 쓰면 CHAR(1)과 동일
VARCHAR(n)		1~65535	• 가변 길이 문자형 저장(variable character의 약자) • n을 1~65535까지 지정
BINARY(n)		1~255	• 고정 길이의 이진 데이터 값 저장
VARBINARY(n)		1~255	• 가변 길이의 이진 데이터 값 저장
TEXT 형식	TINYTEXT	1~255	• 255 크기의 TEXT 데이터 값 저장
	TEXT	1~65535	• N 크기의 TEXT 데이터 값 저장
	MEDIUMTEXT	1~16777215	• 16777215 크기의 TEXT 데이터 값 저장
	LONGTEXT	1~4294967295	• 최대 4GB 크기의 TEXT 데이터 값 저장
BLOB 형식	TINYBLOB	1~255	• 255 크기의 BLOB 데이터 값 저장
	BLOB	1~65535	• N 크기의 BLOB 데이터 값 저장
	MEDIUMBLOB	1~16777215	• 16777215 크기의 BLOB 데이터 값 저장
	LONGBLOB	1~4294967295	• 최대 4GB 크기의 BLOB 데이터 값 저장
ENUM(값들 …)		1 또는 2	• 최대 65535개의 열거형 데이터 값 저장
SET(값들 …)		1, 2, 3, 4, 8	• 최대 64개의 서로 다른 데이터 값 저장

데이터 형

- 날짜와 시간 데이터 형식

데이터 형식	바이트 수	설명
DATE	3	• 'YYYY-MM-DD' 형식으로 날짜 저장 • 저장 범위는 1001-01-01~9999-12-31
TIME	3	• 'HH:MM:SS' 형식으로 시간 저장 • 저장 범위는 -838:59:59.000000~838:59:59.000000
DATETIME	8	• 'YYYY-MM-DD HH:MM:SS' 형식으로 날짜와 시간 저장 • 저장 범위는 1001-01-01 00:00:00~9999-12-31 23:59:59
TIMESTAMP	4	• 'YYYY-MM-DD HH:MM:SS' 형식으로 날짜와 시간 저장 • 저장 범위는 1001-01-01 00:00:00~9999-12-31 23:59:59 • time_zone 시스템 변수와 관련이 있으며 UTC 시간대로 변환하여 저장
YEAR	1	• 'YYYY' 형식으로 연도 저장 • 저장 범위는 1901~2155

데이터 형

- 기타 데이터 형식

데이터 형식	바이트 수	설명
GEOMETRY POINT LINESTRING POLYGON	N/A	• 공간 데이터를 저장하는 형식으로 선, 점, 다각형 같은 공간 데이터 개체를 저장하고 조작
JSON	8	• JSON(JavaScript Object Notation) 문서 저장

데이터를 조작하기 위한 SELECT 문

- SELECT 문
  - SELECT 문은 데이터를 조회하기 위한 SQL 명령어입니다.

형식	SELECT [DISTINCT] {*, column[Alias], . . .} FROM table_name;

  - SQL 명령어는 하나의 문장으로 구성되어야 하는데 여러 개의 절이 모여서 문장이 되는 것이고 이러한 문장들은 반드시 세미콜론(;)으로 마쳐야 합니다.
  - SELECT 문은 반드시 SELECT와 FROM 이라는 2개의 키워드로 구성되어야 합니다.
  - SELECT절은 출력하고자 하는 칼럼 이름을 기술합니다.
  - 특정 컬럼 이름 대신 * 를 기술할 수 있는데, * 는 테이블 내의 모든 컬럼을 출력하고자 할 경우 사용합니다.
  - FROM절 다음에는 조회하고자 하는 테이블 이름을 기술합니다.
  - SQL 문에서 사용하는 명령어들은 대문자와 소문자를 구분하지 않는다는 특징이 있습니다.

데이터를 조작하기 위한 SELECT 문

- SELECT 문
  - 다음은 부서 테이블의 내용을 살펴보기 위한 쿼리문 입니다.

예	SELECT * FROM dept;

  - SELECT는 데이터베이스 내에 저장되어 있는 테이블을 조회하기 위한 명령어입니다.
  - SELECT 다음에는 보고자 하는 대상의 컬럼 이름을 기술합니다. SELECT 다음에 *을 기술하면 지정된 테이블(dept)의 모든 컬럼을 조회합니다.
  - FROM 다음에는 보고자 하는 대상의 테이블 이름을 기술합니다. 위 예에서는 dept를 기술하였기에 dept 테이블에 등록된 부서의 정보를 살펴볼 수 있었습니다.

데이터를 조작하기 위한 SELECT 문

- 특정 데이터만 보기
  - 사원번호에 해당되는 컬럼 이름은 empno이고 사원명에 해당되는 컬럼 이름은 ename입니다. empno와 ename을 출력하기 위해서는 출력하고자 하는 순서대로 기술하되 컬럼과 컬럼 사이에 콤마를 기술합니다.

예	select empno, ename from emp;

데이터를 조작하기 위한 SELECT 문

- 산술 연산자

종류	예
+	**SELECT sal + comm FROM emp;**
-	**SELECT sal - 100 FROM emp;**
*	**SELECT sal * 12 FROM emp;**
/	**SELECT sal / 2 FROM emp;**

- 급여로 연봉 계산을 해보도록 합시다. 일반적으로 연봉은 급여를 12번 곱한 것이므로 연봉을 구하기 위해서 산술 연산자를 사용합시다.

예	SELECT ename, sal, sal*12 FROM emp;

데이터를 조작하기 위한 SELECT 문

- NULL도 데이터이다.
  - 널은 매우 중요한 데이터입니다. 왜냐하면 컬럼에 널 값이 저장되는 것을 허용하는데 널 값을 제대로 이해하지 못한 채 쿼리문을 사용하면 원하지 않는 결과를 얻을 수 있기 때문입니다.
  - 다음은 널에 대한 이해를 돕기 위해서 다양한 널의 정의를 살펴본 것입니다.
    - 0(zero)도 아니고
    - 빈 공간도 아니다.
    - 미확정(해당 사항 없음), 알 수 없는(unknown) 값을 의미한다.
    - 어떤 값인지 알 수 없지만 어떤 값이 존재하고 있다.
    - ? 혹은 ∞의 의미이므로
    - 연산, 할당, 비교가 불가능하다.

데이터를 조작하기 위한 SELECT 문

- NULL도 데이터이다.
  - 널은 ? 혹은 ∞의 의미이기 때문에 연산, 할당, 비교가 불가능합니다.

$$100 + ? = ?$$
$$100 + \infty = \infty$$

  - 다음은 산술 연산자를 이용해서 연봉을 계산하는 쿼리문으로 앞에서 구했던 예제에서 커미션을 연봉 계산에 추가해 본 것입니다. 커미션의 경우에는 널 값을 가진 행도 있으므로 널 값을 가진 데이터와 산술 연산하면 어떤 결과가 나오는지를 확인할 수 있는 좋은 예제입니다.

데이터를 조작하기 위한 SELECT 문

- NULL도 데이터이다.

예	SELECT ename, sal, job, comm, sal*12, sal*12+comm FROM emp;

	ename	sal	job	comm	`sal*12`	`sal*12+comm`
1	SMITH	800	CLERK	<null>	9600	<null>
2	ALLEN	1600	SALESMAN	300	19200	19500
3	WARD	1250	SALESMAN	500	15000	15500
4	JONES	2975	MANAGER	<null>	35700	<null>
5	MARTIN	1250	SALESMAN	1400	15000	16400
6	BLAKE	2850	MANAGER	<null>	34200	<null>
7	CLARK	2450	MANAGER	<null>	29400	<null>
8	SCOTT	3000	ANALYST	<null>	36000	<null>
9	KING	5000	PRESIDENT	<null>	60000	<null>
10	TURNER	1500	SALESMAN	0	18000	18000
11	ADAMS	1100	CLERK	<null>	13200	<null>
12	JAMES	950	CLERK	<null>	11400	<null>
13	FORD	3000	ANALYST	<null>	36000	<null>
14	MILLER	1300	CLERK	<null>	15600	<null>

영업직인 경우 커미션(comm) 컬럼에 값이 저장되어 있으므로 제대로 연봉 계산을 하게 된다.

영업직인 경우 무능력하여 받을 커미션(comm)이 없더라도 0으로 저장되어 있으므로 연봉 계산이 제대로 된다.

영업직인 아닌 경우에는 커미션에 널값이 저장되어 있어서 연봉 계산 결과도 널값으로 구해지는 모순이 발생한다.

데이터를 조작하기 위한 SELECT 문

- NULL도 데이터이다.

예

```
select ename, comm, sal*12+comm,
       ifnull(comm, 0), sal*12+ifnull(comm, 0)
from emp;
```

	ename	comm	`sal*12+comm`	`ifnull(comm, 0)`	`sal*12+ifnull(comm, 0)`
1	SMITH	<null>	<null>	0	9600
2	ALLEN	300	19500	300	19500
3	WARD	500	15500	500	15500
4	JONES	<null>	<null>	0	35700
5	MARTIN	1400	16400	1400	16400
6	BLAKE	<null>	<null>	0	34200
7	CLARK	<null>	<null>	0	29400
8	SCOTT	<null>	<null>	0	36000
9	KING	<null>	<null>	0	60000
10	TURNER	0	18000	0	18000
11	ADAMS	<null>	<null>	0	13200
12	JAMES	<null>	<null>	0	11400
13	FORD	<null>	<null>	0	36000
14	MILLER	<null>	<null>	0	15600

연봉 계산을 위해 사원 테이블에서 급여와 커미션 칼럼을 살펴본 결과 영업사원이 아닌 사원들의 커미션은 NULL로 지정되어 있으므로 연봉을 올바르게 계산하기 위해서는 커미션이 NULL인 경우 0으로 변경하여 계산에 참여하도록 해야 합니다.

오라클에서는 NULL을 0 또는 다른 값으로 변환하기 위해서 사용하는 함수로 IFNULL을 제공합니다. 커미션에 널이 저장되어 있더라도 널을 다른 값으로 변환하는 IFNULL 함수를 사용하면 제대로 된 계산 결과를 얻을 수 있습니다.

데이터를 조작하기 위한 SELECT 문

- 컬럼 이름에 별칭 지정하기
  - SQL에서 쿼리문의 결과가 출력될 때, 컬럼 이름이 컬럼에 대한 헤딩(heading)으로 출력됩니다.

예	select ename, sal*12+ifnull(comm, 0) from emp;

	ename	`sal*12+ifnull(comm, 0)`
1	SMITH	9600
2	ALLEN	19500
3	WARD	15500
4	JONES	35700
5	MARTIN	16400
6	BLAKE	34200
7	CLARK	29400
8	SCOTT	36000
9	KING	60000
10	TURNER	18000
11	ADAMS	13200
12	JAMES	11400
13	FORD	36000
14	MILLER	15600

데이터를 조작하기 위한 SELECT 문

- 컬럼 이름에 별칭 지정하기
  - 컬럼 이름 대신 별칭을 출력하고자 하면 컬럼을 기술한 바로 뒤에 AS 라는 키워드를 쓴 후 별칭을 기술합니다.

예	select ename, sal*12+ifnull(comm, 0) as Annsal from emp;

	ename	Annsal
1	SMITH	9600
2	ALLEN	19500
3	WARD	15500
4	JONES	35700
5	MARTIN	16400
6	BLAKE	34200
7	CLARK	29400
8	SCOTT	36000
9	KING	60000
10	TURNER	18000
11	ADAMS	13200
12	JAMES	11400
13	FORD	36000
14	MILLER	15600

데이터를 조작하기 위한 SELECT 문

- 컬럼 이름에 별칭 지정하기
  - 위 예를 살펴보면 별칭을 부여 할 때에는 대소문자를 섞어서 기술하였는데 출력 결과를 보면 일괄적으로 대문자로 출력된 것을 확인할 수 있습니다.
  - 대소문자를 구별하고 싶으면 “ ” 을 사용합니다.
  - “ ” 을 사용하여 별칭을 부여할 경우에는 별칭에 공백문자나 $,_, #등 특수 문자를 포함시킬 수 있습니다.

예	select ename, sal*12+ifnull(comm, 0) "A n n s a l" from emp;

	ename	`A n n s a l`
1	SMITH	9600
2	ALLEN	19500
3	WARD	15500
4	JONES	35700
5	MARTIN	16400
6	BLAKE	34200
7	CLARK	29400
8	SCOTT	36000
9	KING	60000
10	TURNER	18000
11	ADAMS	13200
12	JAMES	11400
13	FORD	36000
14	MILLER	15600

데이터를 조작하기 위한 SELECT 문

- 컬럼 이름에 별칭 지정하기
  - 영어가 아닌 한글로 별칭을 부여해 봅시다.
  - 오라클에서 한글을 지원하므로 별칭이 아닌 테이블을 생성할 때 컬럼을 설정하면서 컬럼 이름도 한글로 부여할 수 있습니다.

예	select ename, sal*12+ifnull(comm, 0) "연봉" from emp;

	ename	연봉
1	SMITH	9600
2	ALLEN	19500
3	WARD	15500
4	JONES	35700
5	MARTIN	16400
6	BLAKE	34200
7	CLARK	29400
8	SCOTT	36000
9	KING	60000
10	TURNER	18000
11	ADAMS	13200
12	JAMES	11400
13	FORD	36000
14	MILLER	15600

데이터를 조작하기 위한 SELECT 문

- DISTINCT 키워드
  - 다음은 사원들이 소속되어 있는 부서 번호를 출력하기 위한 예입니다.

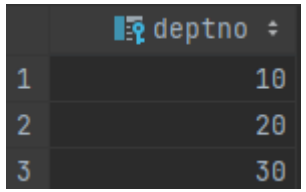
예	select deptno from emp;

	deptno
1	10
2	10
3	10
4	20
5	20
6	20
7	20
8	20
9	30
10	30
11	30
12	30
13	30
14	30

데이터를 조작하기 위한 SELECT 문

- DISTINCT 키워드
  - 사원들이 소속되어 있는 부서의 목록을 얻기 위한 목적이라면 같은 부서의 번호가 중복되어 출력되는 것은 의미가 없습니다.
  - 중복되는 부서 번호를 한 번씩만 출력하기 위해서는 키워드 DISTINCT를 사용합니다.

예	select distinct deptno from emp;



	deptno
1	10
2	20
3	30