DataBase

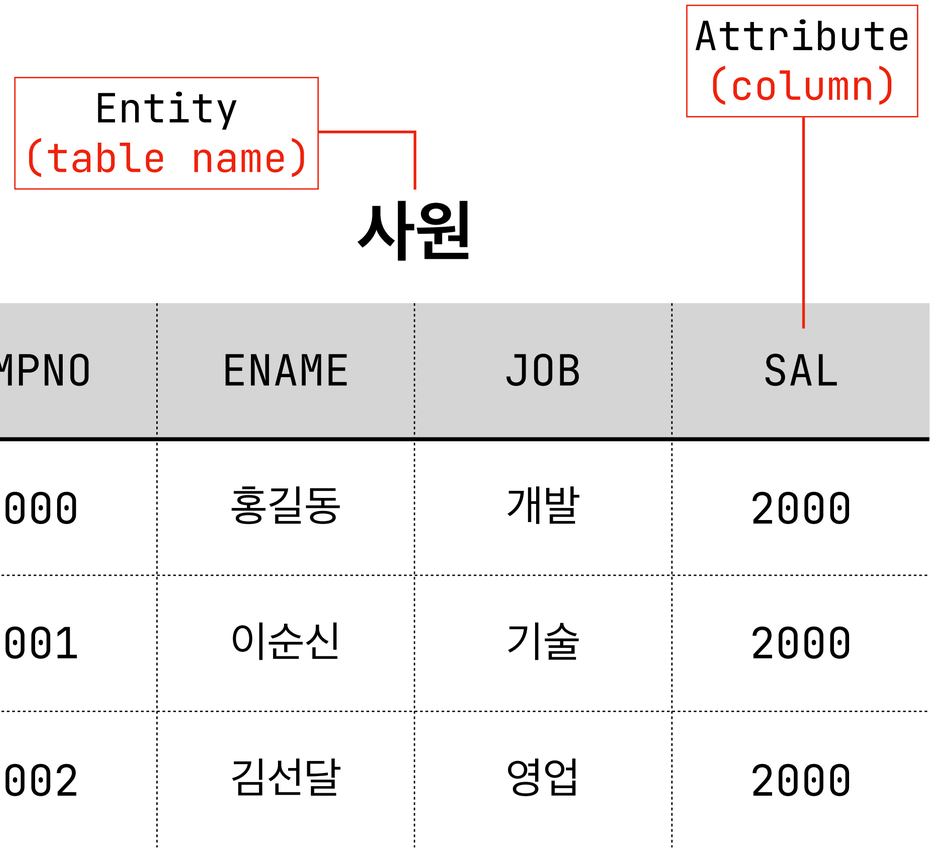
1.database

개념

- Entity 데이터베이스에 저장하고 싶은 개체

- Attribute 해당 개체의 특징

* Schema 메타데이터 집합
  entity
  attribute
  relationship
  ...



1.database

key

- 후보키 (Candidate key) 유일성과 최소성을 만족하는 컬럼
- 기본키 (Primary key) 후보키 중에서 선정되어 사용되는 키
- 대체키 (Alternate key) 후보키 중 기본키로 선정되지 않은 나머지
- 슈퍼키 (Super key) 복합키(Composite key) 또는 연결키
- 외래키 (Foreign key) 관계가 있는 다른 테이블의 컬럼을 식별

1.database

정규화

- 1NF 컬럼의 중복 값 분리
- 2NF 기본키에 종속적이지 않은 컬럼 분리
- 3NF 기본키를 제외한 나머지 컬럼들 간의 종속 불가
- BCNF 기본키가 여러 개 존재할 경우, 복합키로 변경
- 4NF 다치종속 제거 (컬럼 A→B+C : A→B / A→C)
- 5NF 조인 종속성 이용. 조인이 후보키를 통해서만 성립되도록

1.database

transaction

DB를 변화시키기 위한 최소한의 작업 단위

- 원자성 : 모두 반영, 아니면 이전 상태 (All or Nothing)
- 일관성 : 실행 이전과 이후가 일관적이어야 한다
- 고립성 : 실행 도중 다른 트랜잭션의 영향을 받으면 안된다
- 지속성 : 성공하면 결과가 영구적 반영

bonus

oracle type

	자료형	설명
문자	CHAR	최대 길이 2000byte 고정 길이 문자열
	VARCHAR2	최대 길이 4000byte 가변 길이 문자열
	CLOB	최대 4GB 가변 길이 문자열
숫자	NUMBER	숫자형 자료형
날짜	DATE	YYYY-MM-DD HH24:MI:SS
기타	BLOB	이진 데이터 저장 자료형

1.database

sql

Structured Query Language : 구조화된 질의 언어

- Data Definition Language db schema 정의, 조작
- Data Manipulation Language data 조작
- Data Control Language data 제어

1.database

sql

데이터 조작 언어 (Data Manipulation Language)

- SELECT
- INSERT
- UPDATE
- DELETE

1.database

dml

```
SELECT 컬럼명, ...
FROM 테이블명;
```

실행 순서 : FROM → SELECT

ex) 사원 테이블의 전체 목록을 출력하자.

```
SELECT * FROM EMP;
```

Bonus

tip

- Line/Page size 변경

```
set linesize 100; set pagesize 20;
```

- Alias (별칭)

```
SELECT ENAME AS “사원명” FROM EMP;
```

- 테이블 구조 확인

```
DESC {EMP}; (DESCRIBE {EMP};)
```

- 문자열 연결

```
STRING || STRING
```

practice

select

사원 테이블(EMP)의 모든 데이터를 출력하자

사원 테이블에서 사원의 이름(ENAME), 사원번호(EMPNO), 월급(SAL)을 출력하자

사원 테이블에서 사원의 이름과 연봉을 출력하자

사원 테이블에서 사원의 이름, 입사일(HIREDATE), 부서번호(DEPTNO)를 출력하자

사원 테이블에서 사원의 이름과, 해당 사원의 관리자(MGR)를 출력하자

부서 테이블(DEPT)의 구조를 보자

사원 테이블에서 사원의 이름, 월급, 커미션(COMM)을 출력하자

사원 테이블의 모든 데이터를 “ㅇㅇ님이 ㅇㅇ에 입사를 하고 ㅇㅇ의 월급을 받습니다” 형식으로 출력하자
(하나의 컬럼)

1.database

dml

SELECT 컬럼명, ...
FROM 테이블명
WHERE 조건;

실행 순서 : FROM → WHERE → SELECT

연산자 : >, >=, <, <=, =, != (<>), BETWEEN, IN, NOT IN, ANY, ALL, ...

조건의 연결은 AND / OR

practice

select

사원 테이블에서 사원번호가 '7844'인 사원의 사원번호, 이름, 월급을 출력하자

사원 테이블에서 'SMITH'의 사원번호, 이름, 월급을 출력하자

사원 테이블에서 입사일이 1980년 12월 17일인 사원의 모든 데이터를 출력하자

사원 테이블에서 1980년부터 1982년 사이에 입사한 사원의 이름과 입사일을 출력하자

월급이 2000 이하인 사원의 이름과 월급을 출력하자

사원번호가 7369, 7499, 7521인 사원들의 이름과 월급을 출력하자

1.database

dml

```
INSERT INTO 테이블명
VALUES (전체 컬럼 값);

INSERT INTO 테이블명 (특정 컬럼명,…)
VALUES (특정 컬럼 값);
```

사원 테이블에 사원번호:1111, 이름:HONG, 직업:ENGINEER, 관리자:없음, 입사일:오늘 날짜, 월급:3000, 커미션:없음, 부서번호:30을 추가하자

```
INSERT INTO EMP VALUES(1111,'HONG','ENGINEER',NULL,SYSDATE,3000,NULL,30);
```

* SYSDATE : 오늘 날짜 반환

1.database

dml

```
UPDATE 테이블명
SET 컬럼 = 값, 컬럼 = 값, …
WHERE 조건;
```

```
사원테이블에서 HONG의 월급을 3500으로 수정
UPDATE EMP SET SAL = 3500 WHERE ENAME = ‘HONG’;
```

1.database

dml

```
DELETE FROM 테이블명
WHERE 조건;
```

```
사원테이블에서 이름이 HONG인 사원 삭제
DELETE FROM EMP WHERE ENAME = ‘HONG’;
```

1.database

ddl

데이터 정의 언어 (Data Definition Language)

- CREATE
- ALTER
- DROP

1.database

ddl

```
CREATE TABLE 테이블명(
   컬럼명 data_type(size) 제약조건,
   ...
   CONSTRAINT 제약조건명 제약조건 (컬럼명)
)
```

* VIEW : 실제 테이블을 가지고 만든 가상 테이블
(테이블을 조인하여 view 생성시 insert/delete 불가)

```
CREATE OR REPLACE VIEW V_EMP AS SELECT * FROM EMP;
```

1.database

ddl

```
CREATE SEQUENCE 시퀀스명(
   INCREAMENT BY 정수 (기본값 1)   -- 정수값 만큼 증감
   START WITH 정수                 -- 시작 번호
   MAXVALUE 정수                   -- 최대값 지정
   MINVALUE 정수                   -- 최소값 지정
   CYCLE || NOCYCLE                -- 반복 여부
   CACHE 정수 || NOCACHE           -- 정수값 만큼 메모리에 미리 생성
)

SELECT SEQUENCE_TEST.NEXTVAL, SEQUENCE_TEST.CURRVAL FROM DUAL;
```

1.database

ddl

테이블 복제

- 전체 복제

```
CREATE TABLE 새로운 이름 AS SELCT * FROM 테이블 이름
```

- 원하는 컬럼만 복제

```
CREATE TABLE 새로운 이름 AS SELCT 원하는 컬럼 FROM 테이블 이름
```

- 구조(전체 컬럼)만 복제

```
CREATE TABLE 새로운 이름 AS SELCT * FROM 테이블 이름 WHERE 1 = 2;
```

practice

create

size가 10인 문자형 컬럼 ID와 PW를 가진 TEST 테이블을 생성하자

사원 테이블(EMP)의 모든 구조와 데이터를 TEST01로 복사하자

사원 테이블에서 사원의 번호와 이름을 TEST02로 복사하자

사원 테입르에서 사원의 번호와 이름을 TEST03으로 복사하자 (컬럼명을 M1, M2로 변경하자)

사원 테이블의 구조만 TEST04로 복사하자

1.database

ddl

데이터 무결성 : 데이터가 손상되거나 원래의 의미를 잃지 않고 유지되는 상태

제약조건 : 데이터 무결성을 위해 입력되는 자료들의 규칙을 정해줌

제약조건	설명	설정 레벨
NOT NULL	해당 컬럼에 NULL 입력 불가	컬럼
UNIQUE	해당 컬럼 또는 복합컬럼 값이 유일	컬럼, 테이블
PRIMARY KEY	각 행을 유일하게 식별	컬럼, 테이블
CHECK	해당 컬럼의 값이 특정 조건을 항상 만족	컬럼, 테이블

1.database

ddl

제약조건

- 이름으로 관리
  문자로 시작, 길이는 30자까지 가능
  이름을 따로 지정하지 않으면 자동 생성

- 생성 시기
  테이블 생성과 동시 (컬럼 레벨, 테이블 레벨)
  테이블을 생성한 후 (alter table)

1.database

ddl

NOT NULL

```
CREATE TABLE TABLE_NOTNULL01(
  ID CHAR(3) NOT NULL,
  NAME VARCHAR2(20)
)

INSERT INTO TABLE_NOTNULL01 VALUES('100', 'ORACLE')
INSERT INTO TABLE_NOTNULL01(NAME) VALUES('SQLITE')
```

1.database

ddl

NOT NULL

```
CREATE TABLE TABLE_NOTNULL02(
  ID CHAR(3),
  NAME VARCHAR2(20),
  CONSTRAINT TN02_ID_NN NOT NULL(ID)
)
```

1.database

ddl

UNIQUE

```
CREATE TABLE TABLE_UNIQUE01(
  ID CHAR(3) UNIQUE,
  NAME VARCHAR2(20)
)

INSERT INTO TABLE_UNIQUE01 VALUES('100', 'ORACLE');
INSERT INTO TABLE_UNIQUE01 VALUES('200', 'SQLITE');
INSERT INTO TABLE_UNIQUE01 VALUES('200', 'MYSQL');
```

1.database

ddl

UNIQUE

```
CREATE TABLE TABLE_UNIQUE02(
  ID CHAR(3),
  NAME VARCHAR2(20),
  NUM NUMBER,
  CONSTRAINT TU02_ID_UN UNIQUE(ID, NAME)
)

INSERT INTO TABLE_UNIQUE02 VALUES('100', 'ORACLE', 1);
INSERT INTO TABLE_UNIQUE02 VALUES('200', 'SQLITE', 2);
INSERT INTO TABLE_UNIQUE02 VALUES('200', 'MYSQL', 3);
```

1.database

ddl

PRIMATY KEY = UNIQUE + NOT NULL

```
CREATE TABLE TABLE_PK01(
  ID CHAR(3) PRIMARY KEY,
  NAME VARCHAR2(20)
)

INSERT INTO TABLE_PK01 VALUES('100', 'ORACLE');
INSERT INTO TABLE_PK01 VALUES('100', 'SQLITE');
INSERT INTO TABLE_PK01 VALUES(NULL, 'MYSQL');
```

1.database

ddl

PRIMATY KEY = UNIQUE + NOT NULL

```
CREATE TABLE TABLE_PK02(
  ID CHAR(3),
  NAME VARCHAR2(20),
  CONSTRAINT TP02_ID_PK PRIMARY KEY (ID, NAME)
)

INSERT INTO TABLE_PK02 VALUES('100', 'ORACLE');
INSERT INTO TABLE_PK02 VALUES('100', 'SQLITE');
INSERT INTO TABLE_PK02 VALUES(NULL, 'MYSQL');
```

1.database

ddl

FOREIGN KEY

```
CREATE TABLE TABLE_FK01(
  ID CHAR(3),
  NAME VARCHAR2(20),
  PKID CHAR(3) REFERENCES TABLE_PK01(ID)
)

INSERT INTO TABLE_FK01 VALUES('100', 'ORACLE', '100');
INSERT INTO TABLE_FK01 VALUES('100', 'SQLITE', '200');
INSERT INTO TABLE_FK01 VALUES(NULL, 'MYSQL', '300');
```

1.database

ddl

FOREIGN KEY

```
CREATE TABLE TABLE_FK02(
  ID CHAR(3),
  PKID CHAR(3),
  PKNAME VARCHAR2(20),
  FOREIGN KEY(PKID, PKNAME) REFERENCES TABLE_PK02(ID, NAME)
)

INSERT INTO TABLE_FK02 VALUES('100', '100', 'ORACLE');
INSERT INTO TABLE_FK02 VALUES('100', '200', 'SQLITE');
INSERT INTO TABLE_FK02 VALUES(NULL, '300', 'MYSQL');
```

1.database

ddl

CHECK

```
CREATE TABLE TABLE_CHECK01(
  ID CHAR(3),
  NAME VARCHAR2(20),
  MARRIAGE CHAR(1) CHECK (MARRIAGE IN (‘Y’, ‘N’))
)

INSERT INTO TABLE_CHECK01 VALUES(‘100’, ‘HONG’, ‘Y’);
INSERT INTO TABLE_CHECK01 VALUES(‘200’, ‘LEE’, ‘N’);
INSERT INTO TABLE_CHECK01 VALUES(‘300’, ‘KIM’, ‘A’);
```

1.database

ddl

CHECK

```
CREATE TABLE TABLE_CHECK02(
  ID CHAR(3),
  NAME VARCHAR2(20),
  MARRIAGE CHAR(1),
  CONSTRAINT TC02_MRG_CK CHECK (MARRIAGE IN (‘Y’, ‘N’))
)

INSERT INTO TABLE_CHECK02 VALUES(‘100’, ‘HONG’, ‘Y’);
INSERT INTO TABLE_CHECK02 VALUES(‘200’, ‘LEE’, ‘N’);
INSERT INTO TABLE_CHECK02 VALUES(‘300’, ‘KIM’, ‘A’);
```

1.database

ddl

테이블 수정

ALTER TABLE 테이블명

ADD (컬럼명 data_type, ...)
MODIFY (컬럼명 data_type, ...)
DROP COLUMN 컬럼명 || DROP (컬럼명)

TEST 테이블에 SIZE가 20인 문자형 컬럼 ADDR을 추가하자.
ALTER TABLE TEST ADD (ADDR VARCHAR2(20))

1.database

ddl

시퀀스 수정

ALTER SEQUENCE 테이블명

INCREAMENT BY 정수(기본값 1)
MAXVALUE 정수
MINVALUE 정수
CYCLE || NOCYCLE
CACHE 정수 || NOCACHE

* START WITH 값은 수정 불가!

1.database

ddl

테이블 삭제

DROP TABLE 테이블명;

```
TEST 테이블을 삭제하자
DROP TABLE TEST;
```

1.database

dcl

데이터 제어 언어 (Data Control Language)

- COMMIT
- ROLLBACK

TCL (Transaction Control Language)

- GRANT
- REVOKE

1.database

권한

각 사용자에게 권한과 역할을 부여하여 보안 유지

- 시스템 권한
- 객체 권한
- 역할 (ROLE)

1.database

권한

시스템 권한

- 객체 생성, 변경, 소멸 등에 관한 권한으로 SYS(SYSTEM) 에게 부여받음

- 시스템 권한은 기능이 매우 강력하여 대부분 관리자에게만 부여

ex) 테이블스페이스 생성, 임의 테이블 행 삭제 등의 권한

1.database

권한

객체 권한

- 사용자가 특정 객체에 대해 특정 작업을 수행 가능하도록 부여하는 권한

- 객체 내용 조작과 관련된 권한으로, 객체 소유자에게 부여받음

ex) 특정 테이블의 행 삭제

1.database

권한

역할 (ROLE)

- 시스템 권한만 해도 120가지 이상

- 많은 권한을 사용자마다 하나씩 부여하기가 힘듦 → 권한의 집합 (ROLE)

- 사용자에게 특정 ROLE을 부여 (권한 여러개가 한번에)

1.database

권한

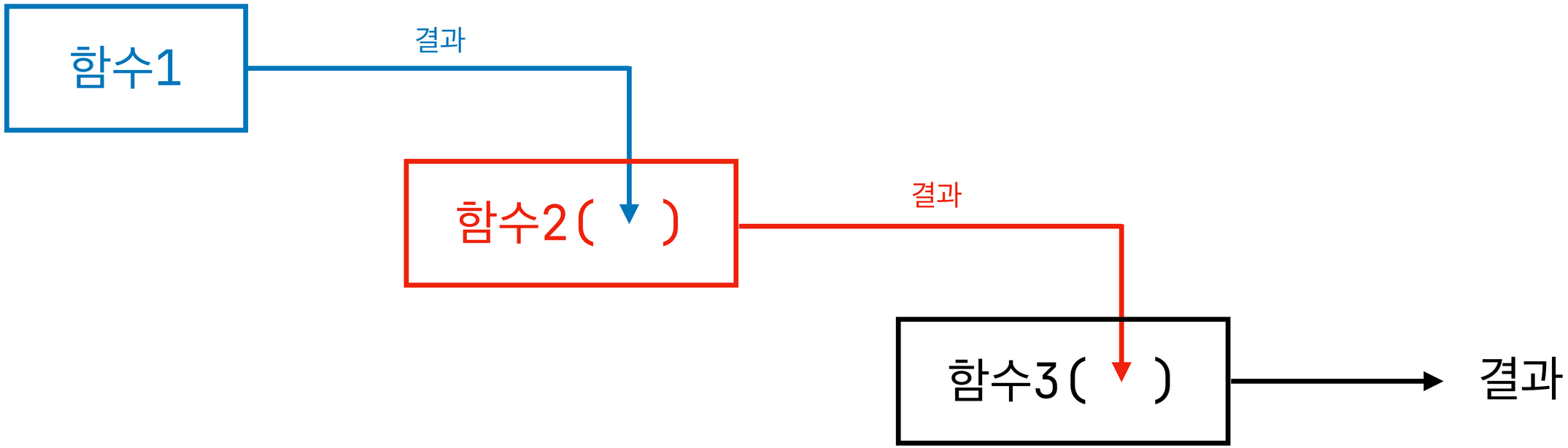
설치와 동시에 기본적으로 생성되어 있는 ROLE

- CONNECT : 접속 권한 등
- RESOURCE : 객체의 생성, 변경, 삭제 등의 기본 시스템 권한 등
- DBA : DB 관리에 필요한 권한 등
- SYSDBA : DB 시작과 종료 및 관리 권한 등
- SYSOPER : SYSDBA + DB 생성 권한 등

2.function

함수 중첩

함수3(함수2(함수1()))



2.function

단일 행 함수

구분	입력 타입	종류	리턴 타입
문자(열)	CHARACTER	LPAD / RPAD, LTRIM / RTRIM / TRIM, SUBSTR	CHARACTER
		INSTR, LENGTH / LENGTHB	NUMBER
숫자	NUMBER	ROUND, TRUNC	NUMBER
날짜	DATE	ADD_MONTHS, SYSDATE	DATE
		MONTHS_BETWEEN	NUMBER
타입 변환	ANY	TO_CHAR, TO_DATE, TO_NUMBER	ANY
기타	ANY	NVL, DECODE	ANY

2.function

단일 행 함수

LPAD / RPAD (컬럼명, 길이, 값)

- 해당 컬럼을 길이만큼 오른쪽 / 왼쪽 정렬한다

- 빈 공간은 값을 채운다

```
사원의 이름을 7자리만큼 오른쪽 정렬하자 (빈 자리는 *)
SELECT LPAD(ENAME, 7, '*') FROM EMP;

사원의 이름을 10자리만큼 왼쪽 정렬하자 (빈 자리는 *)
SELECT RPAD(ENAME, 10, '*') FROM EMP;
```

2.function

단일 행 함수

LTRIM / RTRIM (‘문자열’, ‘제거할 문자(열)’)

- 문자열에서 제거할 문자(열) 제거 → 패턴으로 제거 X

```
‘xyxzyyTech6 327’의 왼쪽에서 xyz를 제거하자
SELECT LTRIM(‘xyxzyyTech6 327’, ‘xyz’) FROM DUAL;

‘xyxzyyTech6 327’의 오른쪽에서 숫자만 제거하자
SELECT RTRIM(‘xyxzyyTech6 327’, ‘0123456789’) FROM DUAL;

‘xyxzyyTech6 327’의 오른쪽에서 공백 및 숫자를 제거하자
SELECT RTRIM(‘xyxzyyTech6 327’, ‘ 0123456789’) FROM DUAL;
```

2.function

단일 행 함수

TRIM(‘제거할 문자 하나’ FROM ‘문자열’)

- 문자열 양쪽에서 제거할 문자 하나 제거

```
‘xyxzyyTech6 327’의 양쪽에서 x를 제거하자
SELECT TRIM(‘x’ FROM ‘xyxzyyTech6 327’) FROM DUAL;

‘xyxzyyTech6 327’의 양쪽에서 xy를 제거하자
SELECT TRIM(‘xy’ FROM ‘xyxzyyTech6 327’) FROM DUAL;
```

2.function

단일 행 함수

SUBSTR(컬럼 or 문자열, 시작위치 [, 반환할 갯수])

- 시작위치부터 [반환할 갯수까지] 문자열 반환

- 시작위치 → 0 or 1 : 처음 / 양수 : 끝 방향으로 / 음수 : 시작 방향으로

- 반환할 갯수 → 음수 : NULL

```
SELECT SUBSTR(ENAME, 2) FROM EMP;
SELECT SUBSTR(ENAME, 2, 4) FROM EMP;
SELECT SUBSTR(ENAME, -2) FROM EMP;
```

2.function

단일 행 함수

INSTR(컬럼 or 문자열, 찾는 문자(열) [, 시작위치 [, 횟수]])

- 찾는 문자(열) 이 [시작위치부터 [횟수만큼]] 나타난 시작위치 반환

- 시작위치 → 양수 : 끝 방향으로 / 음수 : 시작 방향으로

```
SELECT ENAME, INSTR(ENAME, 'S', 1, 1) FROM EMP;
SELECT ENAME, INSTR(ENAME, 'L', -1, 2) FROM EMP;
```

2.function

단일 행 함수

LENGTH / LENGTHB

- 주어진 컬럼의 문자 길이를 반환 (NUMBER / BYTE)

- 컬럼 타입이 CHAR인 경우 데이터의 길이와 상관없이 컬럼 전체 길이 반환

```
SELECT LENGTH(NAME) FROM LENTEST;
SELECT LENGTHB(NAME) FROM LENTEST;

SELECT LENGTH(NAMEB) FROM LENTEST;
SELECT LENGTHB(NAMEB) FROM LENTEST;
```

```
CREATE TABLE LENTEST(
  NAME VARCHAR2(10),
  NAMEB CHAR(10)
)

INSERT INTO LENTEST VALUES('A', 'A');
```

2.function

단일 행 함수

ROUND / TRUNC (컬럼 or 숫자 [, 소수점 자리지정])

- 지정한 자리에서 반올림 / 버림

- 자리 지정하는 값은 반드시 정수 (생략하면 0)
  양수 : 소수점 이하 자리 / 음수 : 소수점 이상 자리

```
SELECT ROUND(123.456) FROM DUAL;
SELECT ROUND(123.456, 1) FROM DUAL;

SELECT TRUNC(123.456, 1) FROM DUAL;
SELECT TRUNC(123.456, -1) FROM DUAL;
```

2.function

단일 행 함수

CEIL / FLOOR()

- 올림 / 버림

```
SELECT CEIL(123.456) FROM DUAL;
SELECT FLOOR(123.456, 1) FROM DUAL;

SELECT CEIL(SAL / 1000) FROM EMP;
SELECT FLOOR(SAL / 1000) FROM EMP;
```

2.function

단일 행 함수

ADD_MONTHS(날짜, 더하려는 개월)

- 지정한 날짜부터 개월수를 더한 날짜 반환

```
입사한 지 20년이 되는 달을 구하자
SELECT ENAME, HIREDATE, ADD_MONTHS(HIREDATE, 240) FROM EMP;
```

2.function

단일 행 함수

MONTHS_BETWEEN(날짜 1, 날짜 2)

- 지정한 두 날짜 사이의 개월수 반환

- 날짜 1 > 날짜 2 : 양수
  날짜 1 < 날짜 2 : 음수

```
00년 1월 1일을 기준으로 10년 이상 근무한 사람의 이름, 직업, 입사일, 근무년을 구하자
SELECT ENAME, JOB, HIREDATE,
       TRUNC(MONTHS_BETWEEN('2000/01/01', HIREDATE)/12)
FROM EMP
WHERE MONTHS_BETWEEN('2000/01/01', HIREDATE) > 120;
```

2.function

단일 행 함수

TO_CHAR, TO_DATE, TO_NUMBER

2.function

단일 행 함수

TO_CHAR(입력 타입 [, 형식])

- 숫자 표현 형식

형식	설명
9	자리 수 지정
0	남는 자리를 0으로 표시
$ or L	통화기호 표시
. or ,	지정한 위치에 . or , 표시

```
SELECT TO_CHAR(1234, ‘99999’) FROM DUAL;
SELECT TO_CHAR(1234, ‘00000’) FROM DUAL;
SELECT TO_CHAR(1234, ‘L9999’) FROM DUAL;
SELECT TO_CHAR(1234, ’9,999’) FROM DUAL;
```

2.function

단일 행 함수

- 날짜 표현 형식

형식	설명
YYYY / YY / YEAR	년도 (4자리 숫자 / 뒤 2자리 숫자 / 문자)
MONTH / MON / MM / RM	달 (이름 / 약어 / 숫자 / 로마기호)
DDD / DD / D	일 (1년 기준 / 1달 기준 / 1주 기준)
Q	분기 (1, 2, 3, 4)
DAY / DY	요일 (이름 / 약어 이름)
HH(12) / HH24	12시간 / 24시간
AM / PM	오전 / 오후
MI	분 (0 ~ 59)
SS	초 (0 ~ 59)

2.function

단일 행 함수

- 날짜 표현 형식

```
SELECT TO_CHAR(SYSDATE, 'HH24:MI:SS') FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'MON DY, YYYY') FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'YYYY-FMMM-DD DAY') FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD') FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'YEAR, Q') FROM DUAL;
```

* FM : 0 or 공백 제거

2.function

단일 행 함수

TO_DATE(입력 타입 [, 형식])

```
SELECT TO_DATE('20100101', 'YYYYMMDD') FROM DUAL;

SELECT TO_CHAR('20100101', 'YYYYMMDD', 'YYYY, MON') FROM DUAL;

SELECT TO_CHAR( TO_DATE('210830 143001', 'YYMMDD HH24MISS'),
                'YY-MM-DD PM FMHH:MI:SS' )
FROM DUAL;
```

2.function

단일 행 함수

TO_NUMBER(입력 타입 [, 형식])

- 변환 시 숫자인 문자열만 가능

```
사원의 이름과, 입사한 년도, 입사한 월을 출력하자
SELECT ENAME,
       TO_NUMBER(SUBSTR(HIREDATE, 1, 2)) AS 년도,
       TO_NUMBER(SUBSTR(HIREDATE, 4, 2)) AS 월
FROM EMP;
```

2.function

단일 행 함수

DECODE(컬럼 or 문자열, 비교값, 같을 때 반환값
[, 비교값, 반환값, ...], [, 다를 때 기본값])

CASE WHEN 비교값 THEN 같을 때 반환값 [WHEN ... THEN …]
[ELSE 다를 때 기본값] END

- ELSE 값을 지정하지 않으면, 조건을 만족시키지 않을 때 NULL 반환

```
SELECT ENAME, JOB, DECODE(JOB, 'MANAGER', '0') FROM EMP;

SELECT ENAME, SAL, CASE WHEN SAL <= 1000 THEN '초급'
       WHEN SAL <= 2000 THEN '중급' ELSE '고급' END
FROM EMP;
```

practice

function

사원 테이블에서 사원 이름의 첫글자는 대문자로, 나머지는 소문자로 변환하여 출력하자

사원 테이블에서 사원 이름, 사원 이름의 두번째 글자부터 네번째 글자까지를 출력하자

사원테이블에서 각 사원 이름의 철자 개수를 출력하자

사원테이블에서 각 사원 이름의 앞 글자 하나와 마지막 글자 하나만 출력하자 (둘 다 소문자로 출력하자)

3456.78 을 소수점 첫번째 자리에서 반올림해서 출력하자 (정수 부분만 출력하자)

Bonus

테이블 연결

- UNION : 합집합 (중복 제거)

```
SELECT DEPTNO FROM DEPT UNION SELECT DEPTNO FROM EMP;
```

- UNION ALL : 합집합 (중복)

```
SELECT DEPTNO FROM DEPT UNION ALL SELECT DEPTNO FROM EMP;
```

- INTERSECT : 교집합

```
SELECT DEPTNO FROM DEPT INTERSECT SELECT DEPTNO FROM EMP;
```

- MINUS : 차집합

```
SELECT DEPTNO FROM DEPT MINUS SELECT DEPTNO FROM EMP;
```

2.function

다중 행 함수

집계함수 (COUNT, MAX, MIN, SUM, AVG)

```
SELECT COUNT(*) FROM EMP;
SELECT COUNT(COMM), COUNT(NVL(COMM, 0)) FROM EMP;

SELECT MAX(SAL), MIN(SAL), SUM(SAL), AVG(SAL) FROM EMP;
```

* 연산에서 NULL은 제외!
* 집계함수에 조건문을 사용 시 → WHERE 대신 HAVING 사용

2.function

다중 행 함수

```
SELECT 컬럼명, ...
FROM 테이블명
WHERE 조건
GROUP BY 컬럼;
```

부서 별 월급 합계를 구하자

```
SELECT DEPTNO, SUM(SAL) FROM EMP GROUP BY DEPTNO;
```

직업별 월급 평균을 구하자

```
SELECT JOB, AVG(SAL) FROM EMP GROUP BY JOB;
```

2.function

다중 행 함수

그룹함수 (ROLLUP, CUBE, GROUPING SET)

- ROLLUP : 순서에 맞게 중간합계 출력

- CUBE : 모든 중간합계 출력

- GROUPING SET : 그룹함수에 원하는 결과를 추가

2.function

다중 행 함수

ROLLUP 출력 순서

```
SELECT A, B, 집계함수 FROM 테이블 GROUP BY ROLLUP(A, B);
```

A, B 에 대한 집계
UNION ALL
A에 대한 집계
UNION ALL
전체 집계

2.function

다중 행 함수

CUBE 출력 순서

SELECT A, B, 집계함수 FROM 테이블 GROUP BY CUBE(A, B);

전체 집계
UNION ALL
B 에 대한 집계
UNION ALL
A 에 대한 집계
UNION ALL
A, B 에 대한 집계

2.function

다중 행 함수

직업, 부서 별 월급의 평균을 출력하자

```
SELECT JOB, DEPTNO, AVG(SAL) FROM EMP
GROUP BY ROLLUP(JOB, DEPTNO);

SELECT JOB, DEPTNO, AVG(SAL) FROM EMP
GROUP BY CUBE(JOB, DEPTNO);

SELECT JOB, DEPTNO, AVG(SAL) FROM EMP
GROUP BY GROUPING SETS(ROLLUP(JOB, DEPTNO), DEPTNO);
```

practice

group by

사원 테이블에서 평균 월급을 출력하자

사원 테이블에서 부서번호가 10인 부서에 근무하고 있는 사원들의 부서번호와 평균 월급을 출력하자

사원 테이블에서 직업이 'SALESMAN'인 사원들의 평균 월급을 출력하자

사원 테이블에서 부서별 평균 월급을 출력하자

사원 테이블에서 직업별 평균 월급을 출력하자

사원 테이블에서 평균 커미션을 출력하자

사원 테이블에서 10번 부서의 최대 월급을 출력하자

사원 테이블에서 부서별 최대 월급을 출력하자

사원 테이블에서 직업별 최대 월급을 출력하자

사원 테이블에서 직업이 'SALESMAN'인 사원들 중 최대월급을 출력하자

2.function

다중 행 함수

```
SELECT 컬럼명, ...
FROM 테이블명
WHERE 조건
ORDER BY 컬럼;
```

- ASC (Default) : 오름차순

- DESC : 내림차순

practice

order by

사원 테이블에서 사원 이름과 월급을 출력하자 (월급을 내림차순으로 출력하자)

사원 테이블에서 직업별 평균 월급을 출력하자 (컬럼 ALIAS를 '평균' 으로 하고, 평균 월급이 높은 순으로 정렬하자)

사원 테이블에서 직업별 총 월급을 출력하자 (총 월급이 5000 이상인 데이터만 출력하자)

사원 테이블에서 부서별 월급의 합을 출력하자 (총 합이 1000 이상인 데이터만 출력하자)

2.function

다중 행 함수

ROWID / ROWNUM

- 식별자로 사용 (수정 불가)

```
CREATE TABLE ROWTEST(NO NUMBER);
INSERT INTO ROWTEST VALUES(111);
INSERT INTO ROWTEST VALUES(222);
INSERT INTO ROWTEST VALUES(333);

SELECT ROWID, ROWNUM, NO FROM ROWTEST;
DELETE FROM ROWTEST WHERE NO = 222;
SELECT ROWID, ROWNUM, NO FROM ROWTEST;
```

2.function

다중 행 함수

Top N Query

월급을 3번째로 많이 받는 사원부터 5번째로 많이 받는 사원의 정보를 전체 출력하자

```
SELECT ENAME, SAL, ROWNUM FROM EMP ORDER BY SAL DESC;

SELECT ENAME, SAL, ROWNUM
FROM (SELECT ENAME, SAL, ROWNUM FROM EMP ORDER BY SAL DESC);

SELECT * FROM
     (SELECT ENAME, SAL, ROWNUM AS RN
     FROM (SELECT ENAME, SAL, ROWNUM FROM EMP ORDER BY SAL DESC) A) B
WHERE RN > 2 AND RN <= 5;
```

2.function

다중 행 함수

순위함수 (RANK, DENSE_RANK, ROW_NUMBER)

- RANK : 동일한 값, 동일한 순위, 동일한 값의 개수만큼 누적순위

- DENSE_RANK : 동일한 값, 동일한 순위, 다음 순위

- ROW_NUMBER : 동일한 값이라도 고유한 순위

```
SELECT ENAME, SAL,
       RANK() OVER (ORDER BY SAL DESC) AS RANK,
       DENSE_RANK() OVER (ORDER BY SAL DESC) AS DENSE,
       ROW_NUMBER() OVER (ORDER BY SAL DESC) AS ROWNMB
FROM EMP;
```

3.sql

subquery

SUBQUERY

- DML 또는 다른 SUBQUERY 안에 들어가는 SELECT

- NESTED SELECT (중첩된 SELECT) 라고도 한다

- single row subquery : 결과가 1개의 row
  multi row subquery : 결과가 여러 개의 row
  multi column subquery : 결과가 여러 개의 column
  inline view : 결과가 가상 테이블로 사용 (from)
  ...

3.sql

subquery

SINGLE ROW SUBQUERY

연산자 : >, =, >=, ...

JONES 보다 더 많은 월급을 받는 사원의 이름과 월급을 출력하자

```
SELECT ENAME, SAL FROM EMP
WHERE SAL > (SELECT SAL FROM EMP WHERE ENAME = ‘JONES’)
```

3.sql

subquery

MULTI ROW SUBQUERY

연산자 : IN, NOT IN, ANY, ALL, ...

부하직원이 없는 사원의 사원번호와 이름을 출력하자

```
SELECT EMPNO, ENAME FROM EMP
WHERE EMPNO NOT IN (SELECT NVL(MGR, 0) FROM EMP)
```

3.sql

subquery

MULTI COLUMN SUBQUERY

직업이 ‘SALESMAN’인 사원과 같은 부서에서 근무하고
같은 월급을 받는 사원들의 이름, 월급, 부서번호를 출력하자

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE (DEPTNO, SAL) IN
      (SELECT DEPTNO, SAL FROM EMP WHERE JOB = ‘SALESMAN’);
```

3.sql

subquery

MULTI COLUMN SUBQUERY

직업이 ‘SALESMAN’인 사원과 같은 부서에서 근무하고
같은 월급을 받는 사원들의 이름, 월급, 부서번호를 출력하자

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE DEPTNO IN
      (SELECT DEPTNO FROM EMP WHERE JOB = ‘SALESMAN’)
AND SAL IN
      (SELECT SAL FROM EMP WHERE JOB = ‘SALESMAN’);
```

3.sql

subquery

INLINE VIEW

자기 부서의 평균 월급보다 더 많은 월급을 받는 사원들의
이름, 월급, 부서번호, 부서별 평균 월급을 출력하자

```
SELECT E.ENAME, E.SAL, MYDEPT.DEPTNO, MYAVG
FROM EMP E,
     (SELECT DEPTNO, AVG(SAL) AS MYAVG
     FROM EMP GROUP BY DEPTNO) MYDEPT
WHERE E.DEPTNO = MYDEPT.DEPTNO
AND E.SAL > MYDEPT.MYAVG;
```

practice

subquery

부서번호가 10번인 사원들과 같은 월급을 받는 사원의 이름과 월급을 출력하자

직업이 ‘CLERK’인 사원들과 같은 부서에서 근무하는 사원의 이름과 월급, 부서번호를 출력하자

‘CHICAGO’에서 근무하는 사원들과 같은 부서에서 근무하는 사원의 이름과 월급을 출력하자

부하직원이 있는 사원의 사원번호와 이름을 출력하자 (자기 자신이 다른 사원의 관리자)

부하직원이 없는 사원의 사원번호와 이름을 출력하자

이름이 ‘KING’인 사원에게 보고하는 사원의 이름과 월급을 출력하자 (관리자가 ‘KING’인 사원)

20번 부서의 사원 중 가장 많은 월급을 받는 사원보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하자

직업이 ‘SALESMAN’인 사원들 중 가장 큰 월급을 받는 사원보다 더 많은 월급을 받는 사원들의 이름과 월급을 출력하자
(MAX 함수를 사용하지 않고 출력하자)

3.sql

join

JOIN : 테이블과 테이블을 연결

- INNER JOIN
- OUTER JOIN
- CROSS JOIN
- NONEQUI JOIN
- SELF JOIN

3.sql

join

```
CREATE TABLE M(
  M1 CHAR(6),
  M2 VARCHAR(10)
);

CREATE TABLE S(
  S1 CHAR(6),
  S2 VARCHAR2(10)
);

CREATE TABLE X(
  X1 CHAR(6),
  X2 VARCHAR2(10)
);
```

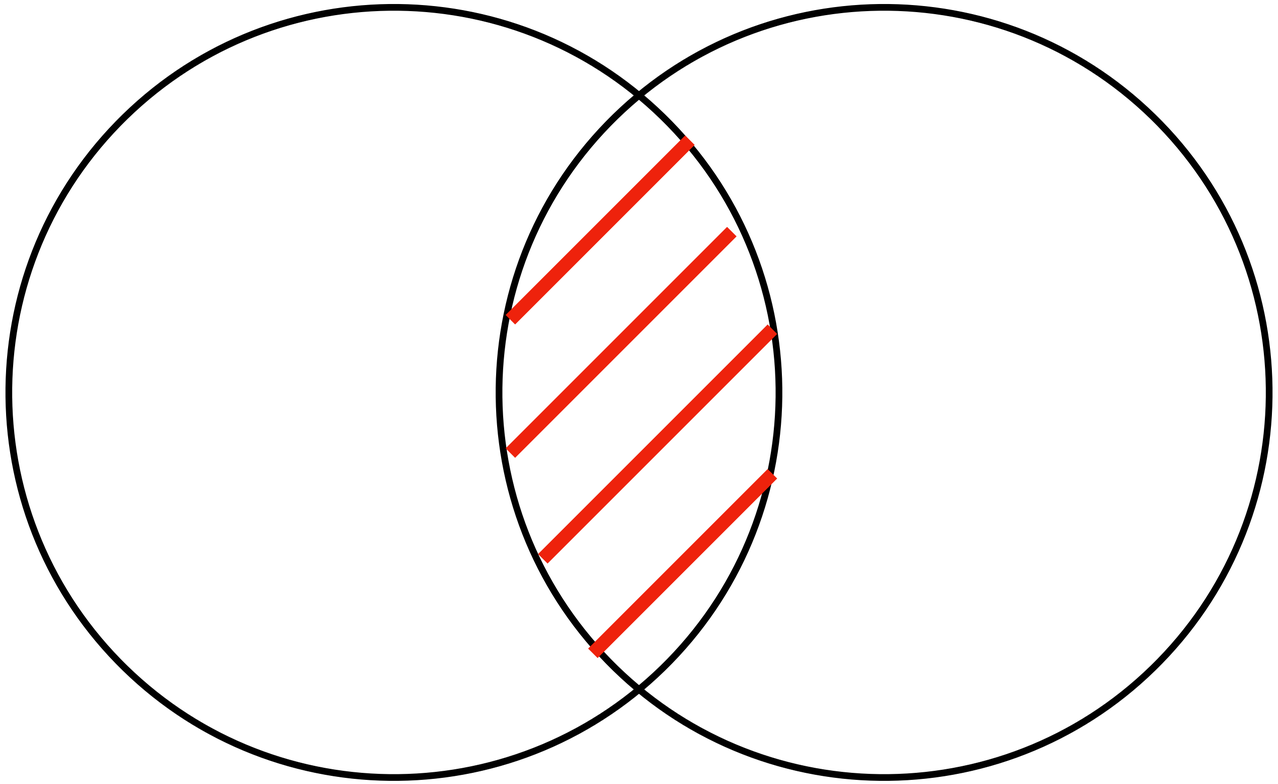
```
INSERT INTO M VALUES('A', '1');
INSERT INTO M VALUES('B', '1');
INSERT INTO M VALUES('C', '3');
INSERT INTO M VALUES(NULL, '3');

INSERT INTO S VALUES('A', 'X');
INSERT INTO S VALUES('B', 'Y');
INSERT INTO S VALUES(NULL, 'Z');

INSERT INTO X VALUES('A', 'DATA');
```

3.sql

join

INNER JOIN

```
SELECT * FROM M INNER JOIN S ON M1 = S1;
```

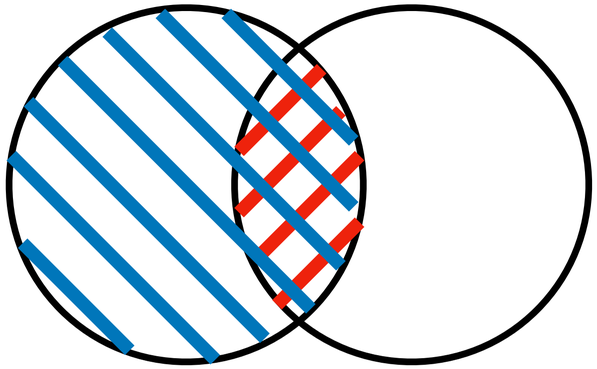


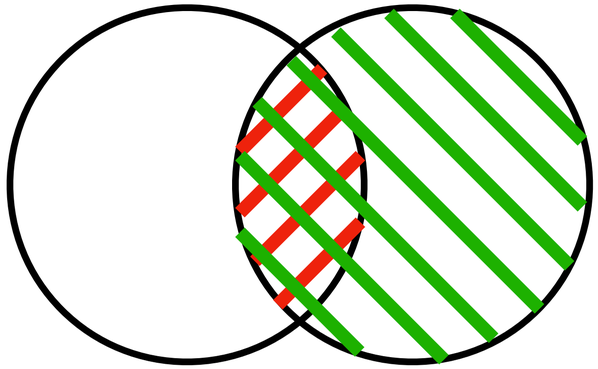
3.sql

join

OUTER JOIN

```
SELECT * FROM M LEFT OUTER JOIN S ON M1 = S1;
SELECT * FROM M ,S WHERE M1 = S1(+);

SELECT * FROM M RIGHT OUTER JOIN S ON M1 = S1;
SELECT * FROM M, S WHERE M1(+) = S1;
```





3.sql

join

CROSS JOIN

```
SELECT * FROM M CROSS JOIN S;
SELECT * FROM M, S;
```

3.sql

join

NON EQUI JOIN

- JOIN 할 테이블의 어떤 컬럼도 일치하지 않을 때

```
SELECT ENAME, SAL, GRADE
FROM EMP JOIN SALGRADE ON (SAL BETWEEN LOSAL AND HISAL);

SELECT ENAME, SAL, GRADE
FROM EMP E, SALGRADE S
WHERE SAL BETWEEN LOSAL AND HISAL;
```

3.sql

join

SELF JOIN

모든 사원의 사번, 이름, 관리자의 사번, 관리자의 이름을 출력하자

```
SELECT 사원.EMPNO, 사원.ENAME, 관리자.EMPNO, 관리자.ENAME
FROM EMP 사원, EMP 관리자
WHERE 사원.MGR = 관리자.EMPNO;
```

practice

join

사원 테이블과 부서 테이블에서 사원들의 이름, 부서번호, 부서이름을 출력하자

사원 테이블과 부서 테이블에서 ‘DALLAS’ 에서 근무하는 사원의 이름, 직업, 부서번호, 부서이름을 출력하자

사원 테이블과 부서 테이블에서 이름에 ‘A’ 가 들어가는 사원들의 이름과 부서이름을 출력하자

사원 테이블과 부서 테이블에서 사원의 이름과 부서의 이름, 월급을 출력하자 (월급이 3000 이산인 사원들만 출력하자)

사원 테이블과 부서 테이블에서 직업이 ‘SALESMAN’인 사원들의 직업과 사원 이름, 부서이름을 출력하자

사원 테이블과 급여 테이블(SALGRADE)에서 커미션이 책정된 사원들의 사원번호, 이름, 연봉, 연봉+커미션, 급여등급(GRADE)을 출력하자
(각각의 컬럼명을 ‘사원번호’, ‘사원이름’, ‘연봉’, ‘연봉2’, ‘급여등급’으로)

사원 테이블과 부서 테이블, 급여 테이블에서 부서번호가 10번인 사원들의 부서번호, 부서이름, 사원이름, 월급, 급여등급을 출력하자
(부서번호가 낮은 순으로, 월급이 높은 순으로 출력하자)

사원 테이블에서 사원번호와 사원이름, 관리자의 사원번호, 사원 이름을 출력하자
(각각의 컬럼명을 ‘사원번호’, ‘사원이름’, ‘관리자번호’, ‘관리자이름’으로)

사원 테이블과 부서 테이블에서 해당 부서의 모든 사원에 대한 부서이름, 위치, 사원수 및 평균 급여를 출력하자

4.ERD

종류

Entity Relationship Diagram

- 바커 표기법 (BARKER NOTATION)

  영국 컨설팅 회사 CACI에 의해 개발, 리차드 바커에 의해 업그레이드

- I/E 표기법 (INFORMATION ENGINEERING NOTATION)

  CLIVE FINKELSTEIN과 JAMES MARTIN이 공동 저술로 발표

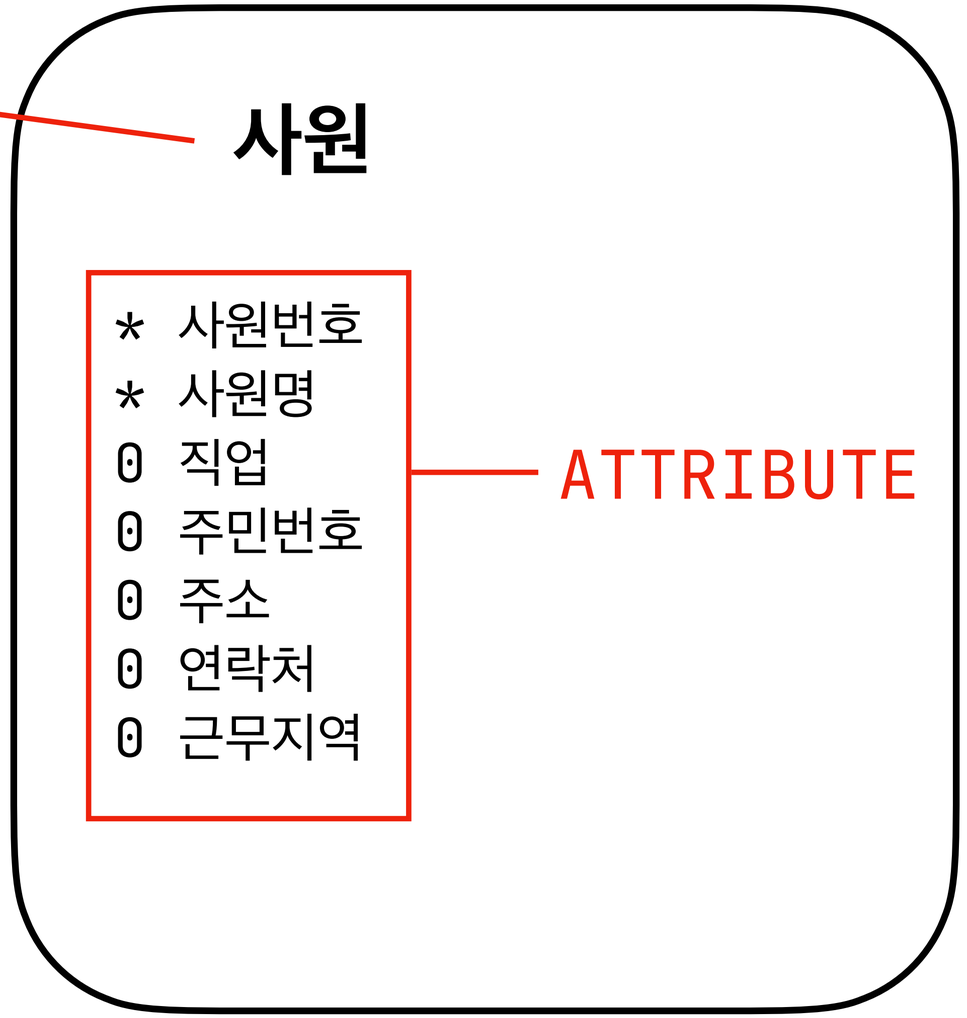
  관계의 다(MANY) 를 나타내기 위해 까마귀 발을 사용 → 까마귀 발 모델이라고도 부른다 (CROW'S FOOT MODEL)

4.ERD

barker

ENTITY

* : not null
0 : null

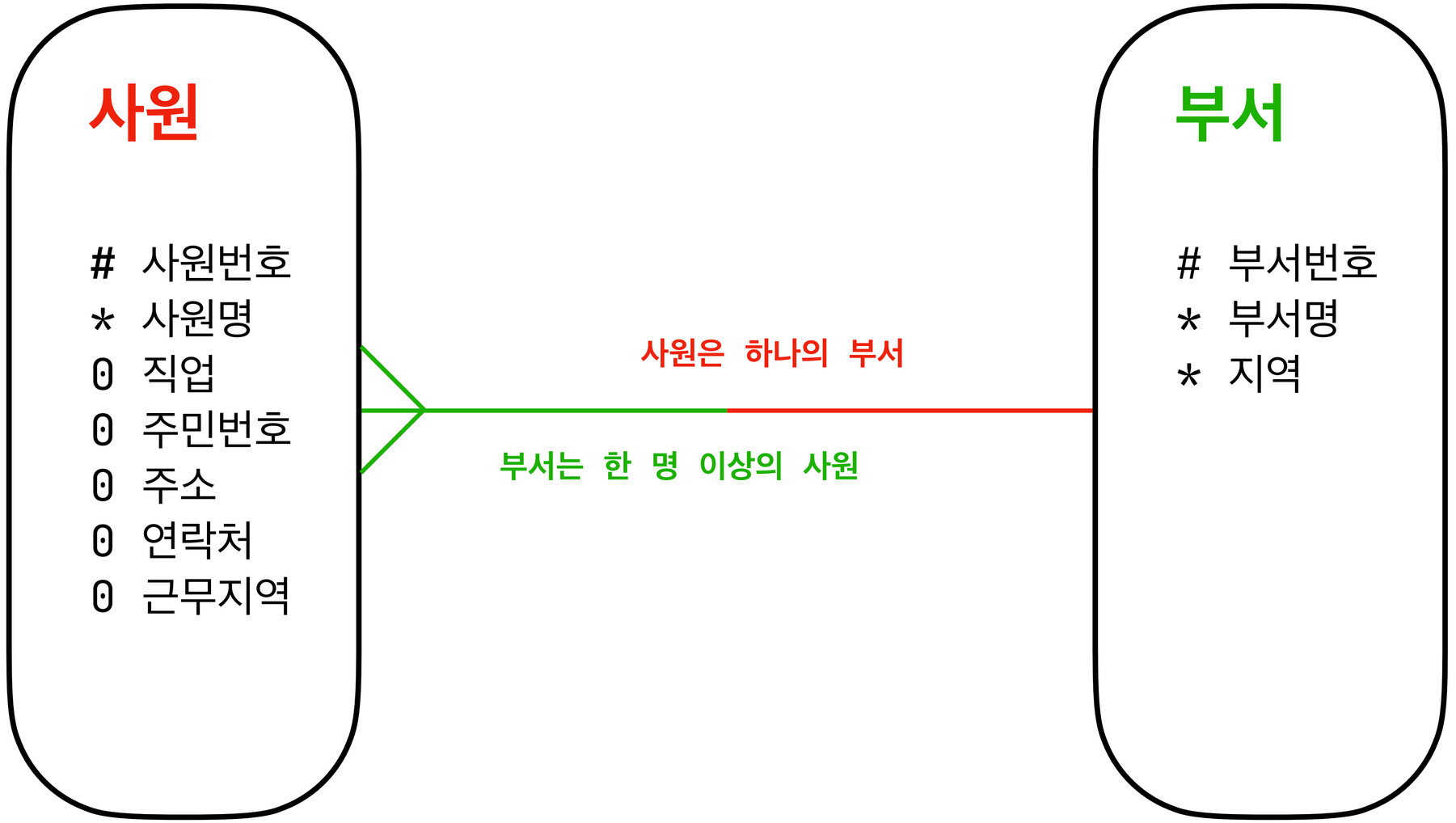


4.ERD

barker

RELATIONSHIP

: primary key



4.ERD

barker

A ⟷ B 의 관계

1 : 1 A 에 존재하는 데이터 1개 ⟷ B 에 존재하는 데이터 1개

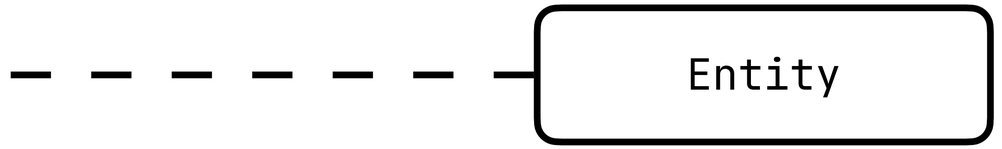
1 : N A 에 존재하는 데이터 1개 ⟷ B 에 존재하는 데이터 N개

N : N A 에 존재하는 데이터 1개 ⟷ B 에 존재하는 데이터 N개
A 에 존재하는 데이터 N개 ⟷ B 에 존재하는 데이터 1개

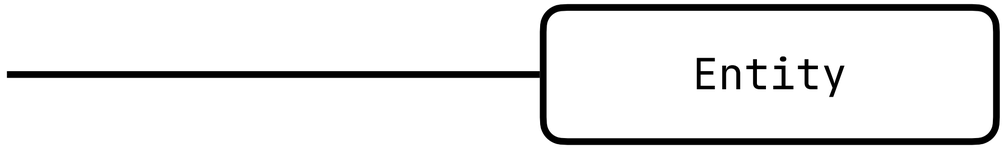
4.ERD

barker

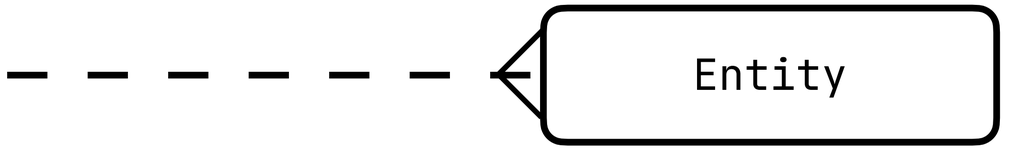
0 or 1



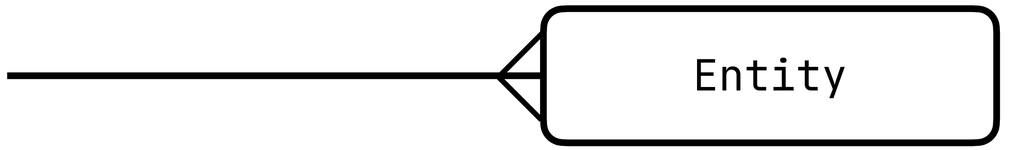
1



0 or 그 이상



1 or 그 이상

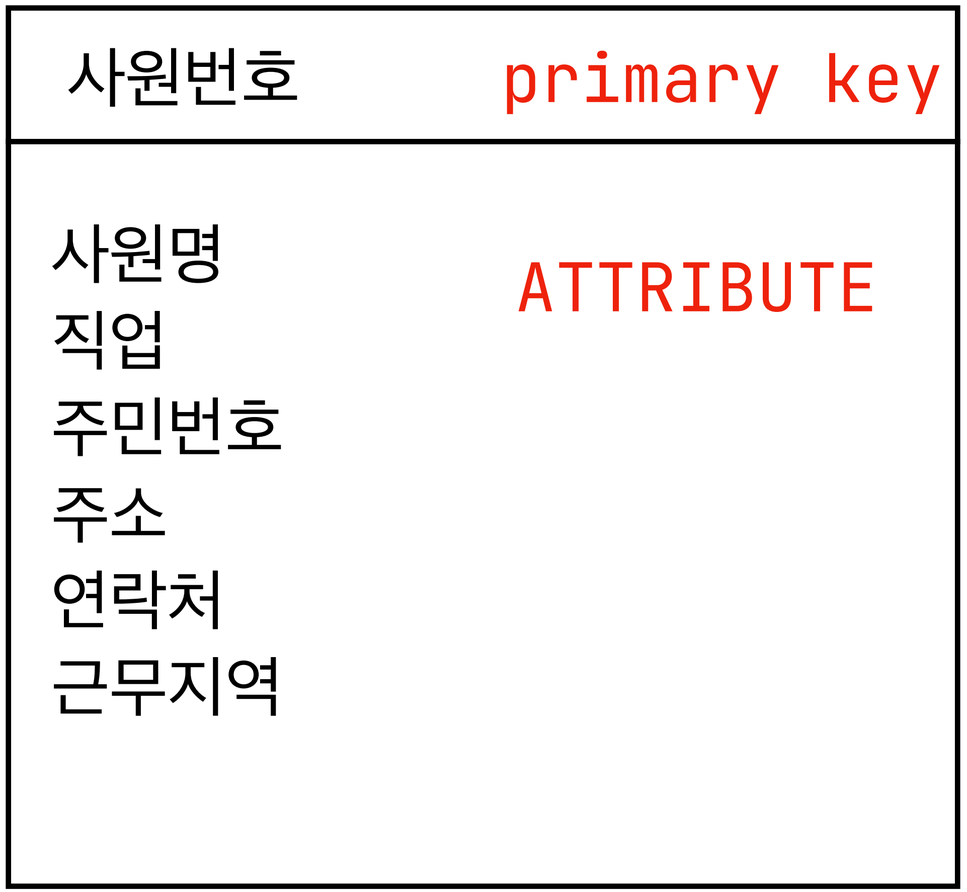


4.ERD

i/e

ENTITY

사원

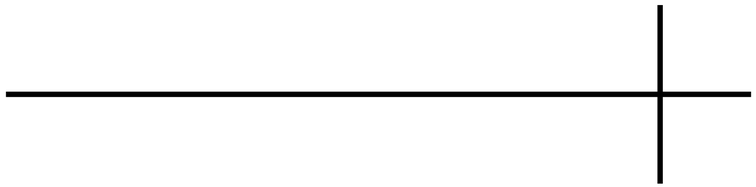


4.ERD

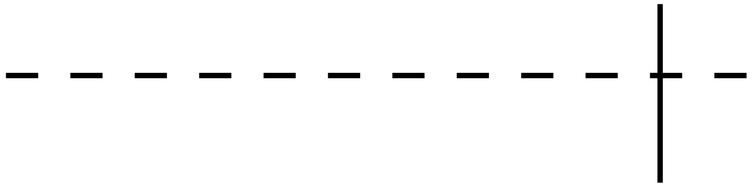
barker

RELATIONSHIP

- IDENTIFYING : A 가 없으면 B 가 존재할 수 없다.



- IDENTIFYING : A 가 없어도 B 가 존재할 수 있다.



4.ERD

barker

0 or 1

1

0 or 그 이상

1 or 그 이상