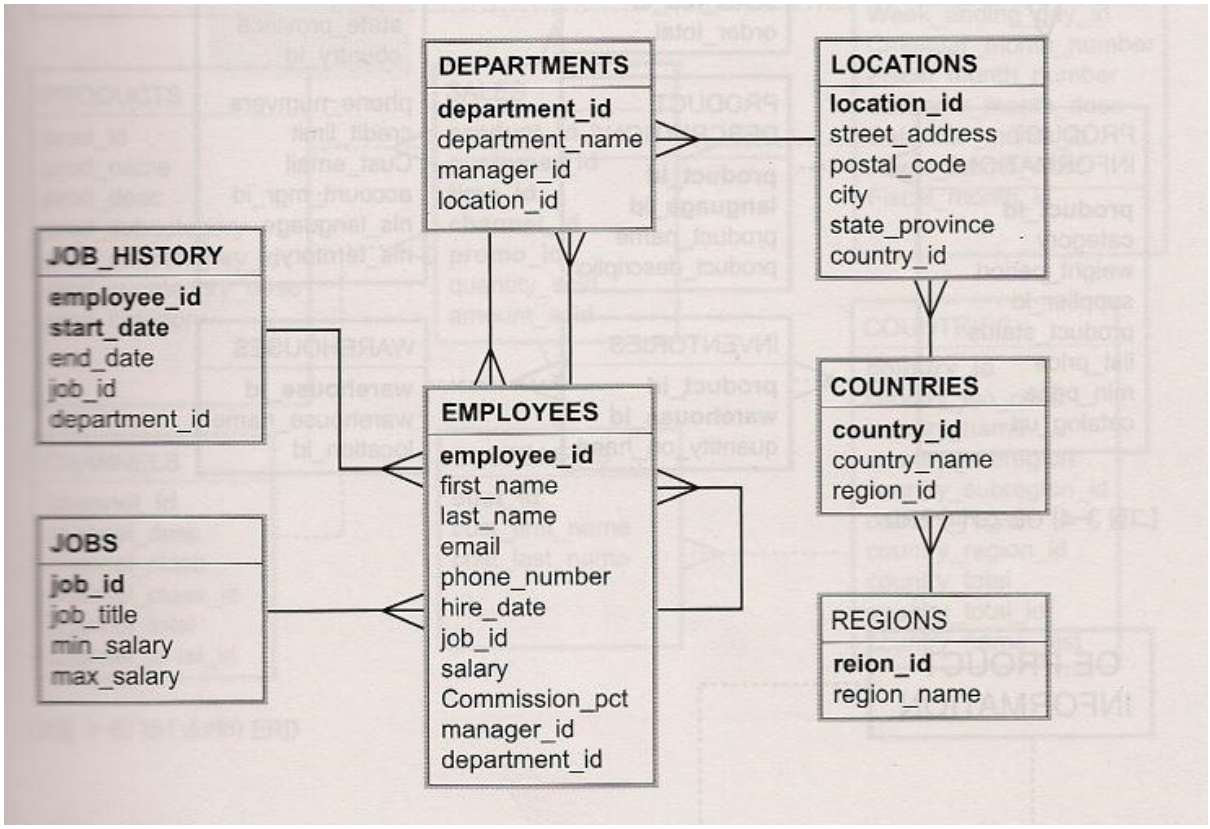
오라클 함수

그룹함수



sum()

- 총합

```
SELECT sum(salary) FROM employees;
```

avg()

- 평균

max()

- 최대

count(*)

- 개수

GROUP BY 절

- 부서별로 평균 급여를 검색

```
SELECT department_id, avg(salary) FROM employees
GROUP BY department_id;
```

주의사항

- SELECT 컬럼 내역은 반드시 그룹함수 또는 GROUP BY절에서 사용한 컬럼

실습 1

- hr > 부서별 사원의 수와 커미션을 받는 사원의 수를 검색

```
SELECT department_id, count(*), count(commission_pct) FROM employees
GROUP BY department_id
ORDER BY department_id;
```

- 화학과 학년별 평균 학점

```
SELECT syear AS 학년, major AS 학과, avg(avr) AS 학점평균 FROM STUDENT
WHERE major = '화학'
GROUP BY major, syear;
```

- 각 학과별 학생수

```
SELECT major "학과", count(*) "학생 수" FROM STUDENT
GROUP BY major
ORDER BY major;
```

- 화학과 생물학과 학생의 4.5 환산 학점의 평균

```
SELECT major "전공", avg(avr * 4.5 / 4.0) "4.5 환산 학점 평균" FROM STUDENT
WHERE major IN('화학', '생물')
GROUP BY major
ORDER BY major;
```

HAVING 절

- 전체 그룹에서 일부 그룹만 추출한 후에 다시 그 그룹에서 어떤 조건을 걸 때 사용
    - WHERE 절을 사용할 수 없음: 이미 알고 있는 정보에 사용 가능, 전체 테이블에서 사용

    - HAVING 절은 알 수 없는 정보에 사용
- 부서별 급여 평균이 5000 미만인 부서의 평균 급여 검색

```
SELECT department_id, AVG(salary)
    FROM employees
    GROUP BY department_id
    HAVING AVG(salary) < 5000;
```

실습 2

- 화학과를 제외한 학생들의 과별 평점 평균

```
SELECT major, ROUND(AVG(avr), 2) FROM STUDENT
WHERE major != '화학'
GROUP BY major;
```

```
SELECT major, ROUND(AVG(avr), 2) FROM STUDENT
GROUP BY major
HAVING major != '화학';
```

- 화학과를 제외한 각 학과별 평점 중 평점이 2.0 이상인 학과 정보 검색

```
SELECT major, AVG(avr) FROM STUDENT
GROUP BY major
HAVING AVG(avr) >= 2.0 AND NOT major = '화학';
```

- 근무 중인 직원이 3명 이상인 부서 검색

```
SELECT count(*), dno FROM emp
GROUP BY dno
HAVING count(*) >= 3;
```

문자함수

LOWER()

- 소문자로 변환

cf) dual: 오라클에서 제공하는 기본 테이블

```
SELECT 'DataBase', LOWER('DataBase') FROM dual;
```

UPPER()

- 대문자로 변환

SUBSTR(문자열, 인덱스, 문자 개수)

- 부분 문자열 추출
- 인덱스는 `1부터` 시작
- 문자 개수는 생략될 수도 있으며, 인덱스는 음수부터 시작하기도 함

```
  SELECT SUBSTR('abcdef', 2, 4) FROM dual; // bcd
```

LENGTH()

- 문자열 길이 추출

실습 3

- kosa01 > 과목명 마지막 글자를 제외하고 출력

```
SELECT cname "과목명", SUBSTR(cname, 1, LENGTH(cname)-1) "실행 결과"
FROM COURSE;
```

LPAD(), RPAD()

- 괄호 안에 문자열, 전체 문자 개수, 채울 문자
- 왼쪽, 오른쪽으로 비어 있는 문자열만큼 채우기

```
SELECT 'Oracle', LPAD('Oracle', 10, '#') FROM dual;
```

숫자 함수

MOD()

- 나머지 연산
- 괄호 안에 나눌 숫자, 몫

ROUND()

- 반올림 연산
- 괄호 안에 소수, 반올림할 자리
- 마이너스 값을 넣으면 정수 값이 반올림됨

날짜 함수

SYSDATE

- 현재 시간 리턴

```
SELECT SYSDATE - 1 "어제", SYSDATE "오늘", SYSDATE + 1 "내일" FROM DUAL;
```

실습 4

- 사원의 근속년 출력 예) 10.5년

```
SELECT first_name "이름", last_name "성", ROUND((SYSDATE - hire_date)/365, 1) || '년'
  FROM EMPLOYEES;
```

- ||: 뒤에 문자열을 붙일 수 있다

변환 함수

TO_CHAR()

- 숫자, 날짜를 문자열로 변환

```
SELECT TO_CHAR(SYSDATE, 'YYYY-MM-DD'),
        TO_CHAR(50000000, '$999,999,999') FROM dual;
```

실습 5

- 2007년도에 입사한 사원의 목록

```
SELECT * FROM employees
    WHERE TO_CHAR(hire_date, 'YYYY') = '2007';
```

TO_DATE()

- 문자열을 날짜 데이터로 변환

```
SELECT TO_DATE('2023-02-10', 'YYYY/MM/DD'),
      TO_DATE('20230211', 'YYYY-MM-DD') FROM dual;
```

NVL()

- NULL을 0 또는 기타 디폴트 값으로 변환

```
SELECT employee_id, salary, NVL(commission_pct, 0) FROM employees;
```

NVL2()

- 값이 NULL인지 아닌지에 따라 각각 다르게 변환

```
SELECT employee_id, salary, NVL2(commission_pct, 'O', 'X') FROM employees;
```

DECODE

- 특정 값에 해당되면 지정해 준 대로 출력

```
SELECT job_id, DECODE(job_id, 'SA_MAN', 'Sales Dept',
              'SH_CLERK', 'Sales Dept', 'Another') FROM employees;
```

CASE WHEN

- 특정 값에 해당되면 지정해 준 대로 출력

```
SELECT job_id
CASE job_id
```

```
WHEN 'SA_MAN' THEN 'Sales Dept'
WHEN 'SH_CLERK' THEN 'Sales Dept'
ELSE 'Another2'
END 'CASE' FROM employees;
```

과제 1, 2 (Scott 계정)

Q1 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요.

EMPNO 열에는 EMP 테이블에서 사원 이름(ENAME)이 다섯 글자 이상이며 여섯 글자 미만인 사원 정보를 출력합니다. MASKING_EMPNO 열에는 사원 번호(EMPNO) 앞 두 자리 외 뒷자리를 * 기호로 출력합니다. 그리고 MASKING_ENAME 열에는 사원 이름의 첫 글자만 보여 주고 나머지 글자 수만큼 * 기호로 출력하세요.

:: 결과 화면

EMPNO	MASKING_EMPNO	ENAME	MASKING_ENAME
7369	73**	SMITH	S****
7499	74**	ALLEN	A****
7566	75**	JONES	J****
7698	76**	BLAKE	B****
7782	77**	CLARK	C****
7788	77**	SCOTT	S****
7876	78**	ADAMS	A****
7900	79**	JAMES	J****

Q2 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요.

EMP 테이블에서 사원들의 월 평균 근무일 수는 21.5일입니다. 하루 근무 시간을 8시간으로 보았을 때 사원들의 하루 급여(DAY_PAY)와 시급(TIME_PAY)을 계산하여 결과를 출력합니다. 단 하루 급여는 소수점 세 번째 자리에서 버리고, 시급은 두 번째 소수점에서 반올림하세요.

:: 결과 화면

EMPNO	ENAME	SAL	DAY_PAY	TIME_PAY
7369	SMITH	800	37.2	4.7
7499	ALLEN	1600	74.41	9.3
7521	WARD	1250	58.13	7.3
7566	JONES	2975	138.37	17.3
7654	MARTIN	1250	58.13	7.3
7698	BLAKE	2850	132.55	16.6
7782	CLARK	2450	113.95	14.2
7788	SCOTT	3000	139.53	17.4
7839	KING	5000	232.55	29.1
7844	TURNER	1500	69.76	8.7
7876	ADAMS	1100	51.16	6.4
7900	JAMES	950	44.18	5.5
7902	FORD	3000	139.53	17.4
7934	MILLER	1300	60.46	7.6

```
// Q1.
SELECT EMPNO, RPAD(SUBSTR(EMPNO, 1, 2), 4, '*') "MASKING_EMPNO",
        ENAME, RPAD(SUBSTR(ENAME, 1, 1), 5, '*') "MASKING_ENMAE"
        FROM EMP
        WHERE LENGTH(ENAME) >= 5 AND LENGTH(ENAME) < 6;

// Q2.
SELECT EMPNO, ENAME, SAL,
TRUNC(SAL/21.5, 2) "DAY_PAY",
ROUND(SAL/(21.5*8), 1) "TIME_PAY"
FROM EMP;
```

Q3 오른쪽과 같은 결과가 나오도록 SQL문을 작성해 보세요.
EMP 테이블에서 사원들은 입사일(HIREDATE)을 기준으로 3개월이 지난 후 첫 월요일에 정직원이 됩니다. 사원들이 정직원이 되는 날짜(R_JOB)를 YYYY-MM-DD 형식으로 오른쪽과 같이 출력해 주세요. 단, 추가 수당(COMM)이 없는 사원의 추가 수당은 N/A로 출력하세요.

:: 결과 화면

EMPNO	ENAME	HIREDATE	R_JOB	COMM
7369	SMITH	1980/12/17	1981-03-23	N/A
7499	ALLEN	1981/02/20	1981-05-25	300
7521	WARD	1981/02/22	1981-05-25	500
7566	JONES	1981/04/02	1981-07-06	N/A
7654	MARTIN	1981/09/28	1982-01-04	1400
7698	BLAKE	1981/05/01	1981-08-03	N/A
7782	CLARK	1981/06/09	1981-09-14	N/A
7788	SCOTT	1987/04/19	1987-07-20	N/A
7839	KING	1981/11/17	1982-02-22	N/A
7844	TURNER	1981/09/08	1981-12-14	0
7876	ADAMS	1987/05/23	1987-08-24	N/A
7900	JAMES	1981/12/03	1982-03-08	N/A
7902	FORD	1981/12/03	1982-03-08	N/A
7934	MILLER	1982/01/23	1982-04-26	N/A

Q4 오른쪽과 같은 결과가 나오도록 SQL문을 작성해 보세요.
EMP 테이블의 모든 사원을 대상으로 직속 상관의 사원 번호(MGR)를 다음과 같은 조건을 기준으로 변환해서 CHG_MGR 열에 출력하세요.

- 직속 상관의 사원 번호가 존재하지 않을 경우 : 0000
- 직속 상관의 사원 번호 앞 두 자리가 75일 경우 : 5555
- 직속 상관의 사원 번호 앞 두 자리가 76일 경우 : 6666
- 직속 상관의 사원 번호 앞 두 자리가 77일 경우 : 7777
- 직속 상관의 사원 번호 앞 두 자리가 78일 경우 : 8888
- 그 외 직속 상관 사원 번호의 경우 : 본래 직속 상관의 사원 번호 그대로 출력

:: 결과 화면

EMPNO	ENAME	MGR	CHG_MGR
7369	SMITH	7902	7902
7499	ALLEN	7698	6666
7521	WARD	7698	6666
7566	JONES	7839	8888
7654	MARTIN	7698	6666
7698	BLAKE	7839	8888
7782	CLARK	7839	8888
7788	SCOTT	7566	5555
7839	KING		0000
7844	TURNER	7698	6666
7876	ADAMS	7788	7777
7900	JAMES	7698	6666
7902	FORD	7566	5555
7934	MILLER	7782	7777

```
// Q3.
```

```
SELECT EMPNO, ENAME, TO_CHAR(HIREDATE, 'YYYY-MM-DD') "HIREDATE",
TO_CHAR(NEXT_DAY(ADD_MONTHS(HIREDATE, 3), '월'), 'YYYY-MM-DD') "R_JOB",
NVL(TO_CHAR(COMM), 'N/A')
FROM EMP;

// Q4.
SELECT EMPNO, ENAME, MGR,
DECODE(TO_CHAR(MGR), NULL, 0000,
EMPNO LIKE '75%', '5555',
EMPNO LIKE '76%', '6666',
EMPNO LIKE '77%', '7777',
EMPNO LIKE '88%, '8888',
MGR) AS CHG_MGR
FROM EMP;
// 다시 해야 됨
```

Q1 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요.
EMP 테이블을 이용하여 부서 번호(DEPTNO), 평균 급여(AVG_SAL), 최고 급여(MAX_SAL), 최저 급여(MIN_SAL), 사원 수(CNT)를 출력합니다. 단 평균 급여를 출력할 때 소수점을 제외하고 각 부서 번호별로 출력하세요.

:: 결과 화면

DEPTNO	AVG_SAL	MAX_SAL	MIN_SAL	CNT
30	1566	2850	950	6
20	2175	3000	800	5
10	2916	5000	1300	3

Q2 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요.
같은 직책(JOB)에 종사하는 사원이 3명 이상인 직책과 인원수를 출력하세요.

:: 결과 화면

JOB	COUNT(*)
CLERK	4
SALESMAN	4
MANAGER	3

Q3 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요.
사원들의 입사 연도(HIRE_YEAR)를 기준으로 부서별로 몇 명이 입사했는지 출력하세요.

:: 결과 화면

HIRE_YEAR	DEPTNO	CNT
1982	10	1
1980	20	1
1981	10	2
1981	20	2
1981	30	6

Q4 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요. 추가 수당(COMM)을 받는 사원 수와 받지 않는 사원 수를 출력하세요.

:: 결과 화면

EXIST_COMM	CNT
X	~~10~~ 8
O	4

Q5 다음과 같은 결과가 나오도록 SQL문을 작성해 보세요. 각 부서의 입사 연도별 사원 수, 최고 급여, 급여 합, 평균 급여를 출력하고 각 부서별 소계와 총계를 출력하세요.

:: 결과 화면

DEPTNO	HIRE_YEAR	CNT	MAX_SAL	SUM_SAL	AVG_SAL
10	1981	2	5000	7450	3725
10	1982	1	1300	1300	1300
10		3	5000	8750	2916.66666666667
20	1980	1	800	800	800
20	1981	2	3000	5975	2987.5
~~20~~	~~1987~~	3 X	~~3000~~	~~7975~~	~~2650~~
20		5	3000	10875	2175
30	1981	6	2850	9400	1566.66666666667
30		6	2850	9400	1566.66666666667
		14	5000	29025	2073.21428571429

과제

- 복습 먼저 우선순위
- 배열 과제, 문자열 과제, for 과제

DDL(정의어)

- 트랜잭션이 적용이 안 됨
  - 한번 적용하면 돌이킬 수 없다

데이터형(자료형)

문자형

- CHAR(size): 고정 → 문자열의 길이가 정해진 경우
- VARCHAR2(size): 가변 → 문자열의 길이를 모를 경우

정수형

- NUMBER

날짜형

- DATE
- TIMESTAMP: 초까지 출력되는 좀 더 정교한 형태

대용량

- LOB
- BLOB
- 공공기관에서 많이 사용한다

CREATE

테이블 생성

```
CREATE TABLE 테이블명 (
  컬럼명1 데이터형,
  컬럼명2 데이터형
)
```

테이블 복사(구조 +데이터)

```
CREATE TABLE 새로 만들 테이블명 (
    AS SELECT * FROM employees;
)
```

테이블 복사(구조만)

```
CREATE TABLE 새로 만들 테이블명 (
    AS SELECT * FROM employees WHERE 1 = 0;
}
```

ALTER (테이블 구조 수정)

- 제약 조건 추가도 ALTER

컬럼 추가

```
ALTER TABLE 테이블명
    ADD (컬럼명 데이터형);
```

컬럼 수정

```
ALTER TABLE 테이블명
    MODIFY(컬럼명 수정내용);
```

컬럼 삭제

```
ALTER TABLE 테이블명
    DROP COLUMN 컬럼명;
```

DELETE

테이블 데이터 삭제

```
// 1. DML: 트랜잭션 대상
DELETE FROM 테이블명;
// 롤백으로 데이터를 다시 살릴 수 있다
ROLLBACK;

// 2. DDL: 트랜잭션 대상 X
TRUNCATE TABLE 테이블명;
```

```
DELETE FROM 테이블 WHERE 삭제대상;
```

실습 8

- dept01 테이블에서 부서 이름 ‘IT Service’ 값을 가진 row 삭제

```
DELETE FROM dept01
WHERE department_name = 'IT Service';
```

DROP

테이블 삭제

```
DROP TABLE 테이블명;
```

실습 6

- departments 테이블로부터 구조와 데이터를 dept01로 복사

```
CREATE TABLE dept01
    AS SELECT * FROM departments;
```

DML

INSERT

데이터 추가

```
// 모든 컬럼의 데이터 값을 넣을 때
INSERT INTO dept01 VALUES(123, 'Jane', 125, 14);

// 일부의 데이터 값 넣을 때 -> 컬럼 지정
INSERT INTO dept01(department_id, department_name)
    VALUES(400, 'SALES');
```

UPDATE

데이터 수정

```
UPDATE 테이블명 SET
바꿀 컬럼명 = '바꿀 값', 컬럼명 = '바꿀 값', ...
WHERE 수정 대상;
```

```
UPDATE dept01 SET
department_name = 'IT Service'
WHERE department_id = 300;
```

실습 7

- emp01 테이블에서 salary 3000 이상 대상자에게 salary 10% 인상

```
UPDATE emp01
SET salary = salary * 1.1
WHERE salary >= 3000;
```

제약 조건

- 데이터베이스 구축(모델링)을 하는 데 필수
- 데이터 추가, 수정, 삭제 가운데 DB 무결성 유지(보장)
  - 데이터의 신뢰를 유지하는 것
  - 데이터 값이 필수무결하게 내가 필요한 그 값인지를 보장하는 것
- 테이블 생성

```
CREATE TABLE emp01 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER
)
```

- 테이블에 내용 삽입
  - null 값 방지가 안 된다

```
INSERT INTO emp01 VALUES(null, null, 'IT', 30);
```

- 테이블에 `NOT NULL` 조건으로 생성
  - 중복 값 방지가 안 된다

```
CREATE TABLE emp02 (
  empno NUMBER NOT NULL,
  ename VARCHAR2(20) NOT NULL,
  job VARCHAR2(20),
  deptno NUMBER
)

INSERT INTO emp02 VALUES(null, null, 'IT', 30);
// ORA-00922: missing or invalid option
// NULL 값 방지
INSERT INTO emp02 VALUES(100, 'kim', 'IT', 30);
INSERT INTO emp02 VALUES(100, 'park', 'IT', 30);
```

- 테이블을 `UNIQUE` 조건으로 생성

```
CREATE TABLE emp03 (
  empno NUMBER UNIQUE,
  ename VARCHAR2(20) NOT NULL,
  job VARCHAR2(20),
  deptno NUMBER
)

NSERT INTO emp02 VALUES(100, 'kim', 'IT', 30);
INSERT INTO emp02 VALUES(100, 'park', 'IT', 30);
// ORA-00001: unique constraint (HR.SYS_C007020) violated
// 중복값 방지
```

`PRIMARY KEY` (기본키, 식별자)

- UNIQUE + NOT NULL

```
CREATE TABLE emp04 (
  empno NUMBER PRIMARY KEY,
  ename VARCHAR2(20) NOT NULL,
  job VARCHAR2(20),
  deptno NUMBER
)

// 300까지밖에 데이터가 없는데 이렇게 넣으면 무결성 조건 위반
INSERT INTO emp04 VALUES(100, 'park', 'IT', 3000);
```

FOREIGH KEY (외래키)

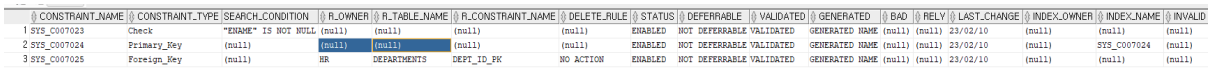
- 데이터 범위에 벗어나는 값 방지

column level 방식

- column 차원에서 제약 조건 생성

```
CREATE TABLE emp05 (
  empno NUMBER PRIMARY KEY,
  ename VARCHAR2(20) NOT NULL,
  job VARCHAR2(20),
  // department_id 데이터 값을 참조하겠다
  deptno NUMBER REFERENCES departments(department_id)
)

INSERT INTO emp05 VALUES(150, 'park', 'IT', 3000);
// ORA-02291: integrity constraint (HR.SYS_C007025) violated - parent key not found
// 데이터 범위에 벗어나는 값 방지
```



	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATUS	DEFERRABLE	VALIDATED	GENERATED	BAD	RELY	LAST_CHANGE	INDEX_OWNER	INDEX_NAME	INVALID
1	SYS_C007023	Check	"ENAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)	(null)	23/02/10	(null)	(null)	(null)
2	SYS_C007024	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)	(null)	23/02/10	(null)	SYS_C007024	(null)
3	SYS_C007025	Foreign_Key	(null)	HR	DEPARTMENTS	DEPT_ID_PK	NO ACTION	ENABLED	NOT DEFERRABLE	VALIDATED	GENERATED NAME	(null)	(null)	23/02/10	(null)	(null)	(null)

- 시스템이 만든 이름으로 제약 조건이 만들어짐

table level 방식

```
CREATE TABLE emp06 (
  empno NUMBER,
  ename VARCHAR2(20) NOT NULL,
  job VARCHAR2(20),
  // department_id 데이터 값을 참조하겠다
  deptno NUMBER,

  CONSTRAINT emp06_empno_pk PRIMARY KEY(empno),
  CONSTRAINT emp06_deptno_fk
      FOREIGN KEY(deptno)
      REFERENCES department(departments_id)
);
```

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE	STATU
1	EMP06_DEPTNO_FK	Foreign_Key	(null)	HR	DEPARTMENTS	DEPT_ID_PK	NO ACTION	ENABLED
2	EMP06_EMPNO_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)	ENABLED
3	SYS_C007026	Check	"ENAME" IS NOT NULL	(null)	(null)	(null)	(null)	ENABLED

- 제약 조건 이름이 내가 설정한 이름으로 명시됨

테이블 수정 방식

- `alter` 이용
- NOT NULL, UNIQUE는 `MODIFY` 이용

```
CREATE TABLE emp07 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER
);

ALTER TABLE emp07
  ADD CONSTRAINT emp07_empno_pk PRIMARY KEY(empno);

ALTER TABLE emp07
  ADD CONSTRAINT emp07_deptno_fk FOREIGN KEY(deptno)
  REFERENCES departments(department_id);

// NOT NULL은 MODIFY 이용
ALTER TABLE emp07
  MODIFY ename CONSTRAINT emp07_ename_nn NOT NULL;
```

```
CREATE TABLE emp08 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER
);

ALTER TABLE emp08
    ADD CONSTRAINT emp08_empno_pk PRIMARY KEY(empno)
    ADD CONSTRAINT emp08_deptno_fk FOREIGN KEY(deptno)
        REFERENCES departments(department_id)
    MODIFY ename CONSTRAINT emp08_ename_nn NOT NULL;
```

IN

```
CREATE TABLE emp09 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER,
  -- F와 M 둘 중 하나만 가능
  gender CHAR(1) CHECK(gender IN('F', 'M'))
);

INSERT INTO emp09 VALUES(100, 'park', 'IT', 20, 'A');
-- 제약 조건 위반
-- ORA-02290: check constraint (HR.SYS_C007032) violated
```

DEFAULT 제약 조건

```
CREATE TABLE emp10 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER,
  -- 생략해도 기본값이 Seoul
  loc VARCHAR2(20) DEFAULT 'Seoul'
);

INSERT INTO emp10(empno, ename, job, deptno)
      VALUES(100, 'kim', 'IT', 30);
```

2개 이상 식별자 설정

- 컬럼 레벨 방식에서는 불가능

```
CREATE TABLE emp11 (
  empno NUMBER PRIMARY KEY,
  ename VARCHAR2(20) PRIMARY KEY,
  job VARCHAR2(20),
  deptno NUMBER,
  loc VARCHAR2(20) DEFAULT 'Seoul'
);
```

- 주 키(Primary Key)가 두 개면 두 값 모두 중복일 때만 걸러짐

```
CREATE TABLE emp12 (
  empno NUMBER,
  ename VARCHAR2(20),
  job VARCHAR2(20),
  deptno NUMBER,
  loc VARCHAR2(20) DEFAULT 'Seoul'
);

ALTER TABLE emp12
  ADD CONSTRAINT emp12_no_name_pk PRIMARY KEY(empno, ename);

INSERT INTO emp12 VALUES(100, 'park', 'IT', 20);
INSERT INTO emp12 VALUES(100, 'kim', 'IT', 20);
```

실습 9

- dept01 테이블의 department_id column를 기본키 제약 조건 구현
- emp13 테이블의 deptno column이 dept01 테이블의 department_id를 참조하도록 구현

- 테이블 수정 방식

```
CREATE TABLE emp13 (
    empno NUMBER PRIMARY KEY,
    ename VARCHAR2(20),
    job VARCHAR2(20),
    deptno NUMBER,
    loc VARCHAR2(20) DEFAULT 'Seoul'
);

ALTER TABLE dept01
    ADD CONSTRAINT dept01_dptid_pk PRIMARY KEY(department_id);

ALTER TABLE emp13
    ADD CONSTRAINT emp13_deptno_fk FOREIGN KEY(deptno)
    REFERENCES dept01(department_id);

INSERT INTO emp13 VALUES(100, 'park', 'IT', 3000);
```

제약 조건 삭제(해제)

```
ALTER TABLE 테이블명
    DROP CONSTRAINT 제약조건명;

INSERT INTO emp13 VALUES(100, 'park', 'IT', 30);

// 사람이 있어서 삭제가 안 된다
DELETE FROM dept01 WHERE department_id = 30;
```

해결 방법

1. emp13 테이블의 데이터 중 deptno = 30 값을 가진 행을 삭제한 후 dept01 테이블의 30번 부서 삭제

```
DELETE FROM emp13 WHERE deptno = 30;
DELETE FROM emp13 WHERE department_id = 30;
```

2. 테이블 설정 시부터 CASCADE 이용해 부서 테이블이 삭제되면 참조하는 데이터도 함께 삭제

```
CREATE TABLE emp14 (
    empno NUMBER PRIMARY KEY,
    ename VARCHAR2(20) NOT NULL,
    job VARCHAR2(20),
    deptno NUMBER REFERENCES dept01(department_id)
```

```
    // 참조하는 내용이 삭제되면 같이 삭제
    ON DELETE CASCADE
);

INSERT INTO emp14 VALUES(100, 'park', 'IT', 20);
```

제약 조건 과제

- 테이블 수정 방식

1. 회원 정보를 저장하는 테이블을 MEMBER란 이름으로 생성한다.

컬럼명	자료형	크기	유일키	NULL허용	키	비고
ID	VARCHAR2	20	Y	N	PK	회원ID
NAME	VARCHAR2	20	N	N		이름
REGNO	VARCHAR2	13	Y	N		주민번호
HP	VARCHAR2	13	Y	Y		핸드폰번호
ADDRESS	VARCHAR2	100	N	Y		주소

2.도서정보를 저장하는 테이블 BOOK이라는 이름을 생성한다.

컬럼명	자료형	크기	유일키	NULL허용	키	비고
CODE	NUMBER	4	Y	N	PK	제품코드
TITLE	VARCHAR2	50	N	N		도서명
COUNT	NUMBER	6	N	Y		수량
PRICE	NUMBER	10	N	Y		정가
PUBLISH	VARCHAR2	50	N	Y		출판사

3. 회원이 책을 주문하였을 때 이에 대한 정보를 저장하는 테이블 이름은 ORDER2로한다.

컬럼명	자료형	크기	유일키	NULL허용	키	비고
NO	VARCHAR2	10	Y	N	PK	주문번호
ID	VARCHAR2	20	N	N	FK	회원ID
CODE	NUMBER	4	N	N	FK	제품번호
COUNT	NUMBER	6	N	Y		주문건수
DR_DATE	DATE		N	Y		주문일자

```
-- 1.
CREATE TABLE MEMBER2 (
    ID VARCHAR2(20),
    NAME VARCHAR2(20),
    REGNO VARCHAR2(13),
    HP VARCHAR2(13),
    ADDRESS VARCHAR(100)
);
```

```
ALTER TABLE MEMBER2
    ADD CONSTRAINT member2_id_pk PRIMARY KEY(ID)
    MODIFY NAME CONSTRAINT member2_name_nn NOT NULL
    MODIFY REGNO CONSTRAINT member2_regno NOT NULL
    MODIFY REGNO CONSTRAINT mem2_regno_un UNIQUE;
    MODIFY HP CONSTRAINT mem2_hp UNIQUE;

-- 2.
CREATE TABLE BOOK (
    CODE NUMBER(4),
    TITLE VARCHAR2(50),
    COUNT NUMBER(6),
    PRICE NUMBER(10),
    PUBLISH VARCHAR2(50)
);

ALTER TABLE BOOK
    ADD CONSTRAINT book_code_pk PRIMARY KEY(CODE)
    MODIFY TITLE CONSTRAINT book_title_nn NOT NULL;

-- 3.
CREATE TABLE ORDER2 (
    NO VARCHAR2(10),
    ID VARCHAR2(20),
    CODE NUMBER(4),
    COUNT NUMBER(6),
    DR_DATE DATE
);

ALTER TABLE ORDER2
    ADD CONSTRAINT order2_no_pk PRIMARY KEY(no);
    ADD CONSTRAINT order2_id_fk FOREIGN KEY(ID)
        REFERENCES MEMBER2(id)
    ADD CONSTRAINT order2_code_fk FOREIGN KEY(code)
        REFERENCES BOOK(code)
    MODIFY id CONSTRAINT order2_id_nn NOT NULL
    MODIFY code CONSTRAINT order2_code_nn NOT NULL;
```

제약 조건 과제 2

DEPT_CONST 테이블과 EMP_CONST 테이블을 다음과 같은 특성
및 제약조건을 지정하여 만들어 보세요.

DEPT_CONST 테이블

컬럼이름	자료형	길이	제약조건	제약조건이름
DEPTNO	정수형	2	PRIMARY KEY	DEPTCONST_DEPTNO_PK
DNAME	문자형	14	UNIQUE	DEPTCONST_DNAME_UNQ
LOC	문자형	13	NOT NULL	DEPTCONST_LOC_NN

EMP_CONST테이블

컬럼이름	자료형	길이	제약조건	제약조건이름
EMPNO	정수형	4	PRIMARY KEY	EMPCONST_EMPNO_PK
ENAME	문자형	10	NOT NULL	EMPCONST_ENAME_NN
JOB	문자형	9		
TEL	문자형	20	UNIQUE	EMPCONST_TEL_UNQ
HIREDATE	날짜			
SAL	정수형	7	CHECK:1000~9999	EMPCONST_SAL_CHK
COMM	정수형	7		
DEPTNO	정수형	2	FOREIGN KEY	EMPCONST_DEPTNO_FK

```
CREATE TABLE DEPT_CONST (
    DEPTNO NUMBER(2),
    DNAME VARCHAR2(14),
    LOC VARCHAR2(13)
);

ALTER TABLE DEPT_CONST
    ADD CONSTRAINT deptconst_deptno_pk PRIMARY KEY(DEPTNO)
    MODIFY DNAME CONSTRAINT deptconst_dname_unq UNIQUE
    MODIFY LOC CONSTRAINT deptconst_loc_nn NOT NULL;

CREATE TABLE EMP_CONST (
    EMPNO NUMBER(4),
    ENAME VARCHAR2(10),
    JOB VARCHAR2(9),
    TEL VARCHAR2(20),
    HIREDATE DATE,
    SAL NUMBER(7),
    COMM NUMBER(7),
    DEPTNO NUMBER(2)
);

ALTER TABLE emp_const
    ADD CONSTRAINT empconst_empno_pk PRIMARY KEY(EMPNO)
    MODIFY ENAME CONSTRAINT empconst_ename_nn NOT NULL
    MODIFY TEL CONSTRAINT empconst_tel_unq UNIQUE
    ADD CONSTRAINT empconst_sal_chk CHECK(SAL >= 1000 AND SAL <= 9999)
    ADD CONSTRAINT empconst_deptno_fk FOREIGN KEY(DEPTNO)
        REFERENCES dept_const(deptno);
```