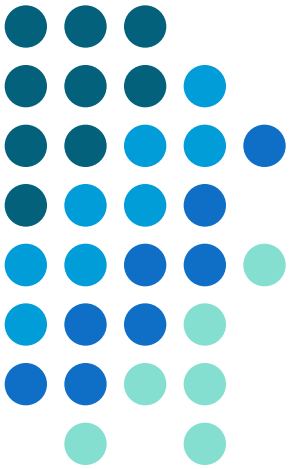
1장 Database란?



데이터베이스의 개요

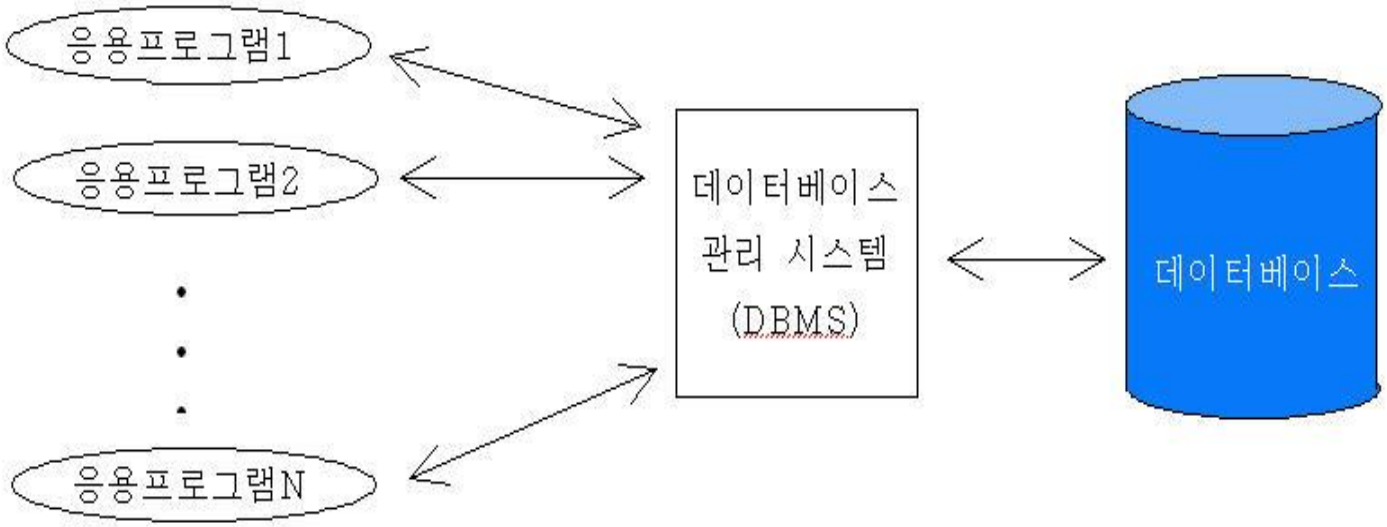
- 데이터베이스는 유용한 데이터의 집합
  - 검색에 용이하게 데이터를 저장하고
  - 수정과
  - 삭제가 용이해야 한다.

파일시스템

데이터를 가공, 처리하여
유용한 정보를 얻기 위한
기본적인 데이터 저장 도구로
초기에 사용된 것

DBMS

- 응용 프로그램과 데이터베이스의 중재자로서 모든 응용 프로그램들이 데이터베이스를 공용할 수 있게 끔 관리해 주는 소프트웨어 시스템



DBMS의 장점

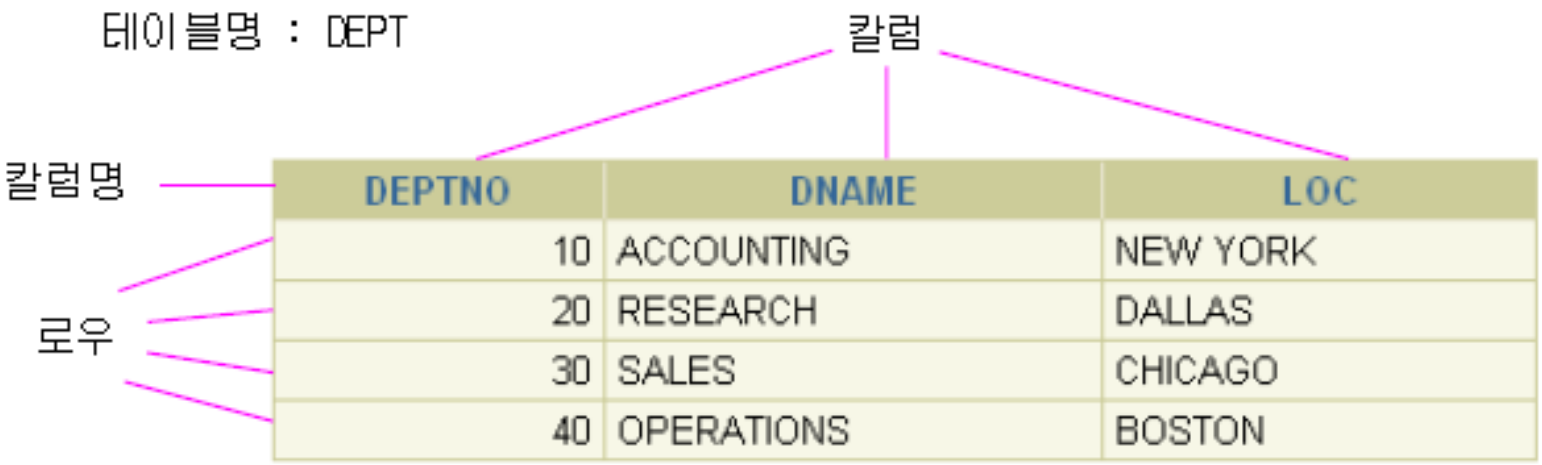
- 데이터의 공유가 가능하다.
- 데이터 중복성이 감소된다.
- 데이터 불일치를 피할 수 있다.
- 데이터의 무결성을 유지할 수 있다.
- 데이터 보안을 유지할 수 있다.
- 표준화가 가능하다.

관계형 DBMS

- Relational Data Base Management System

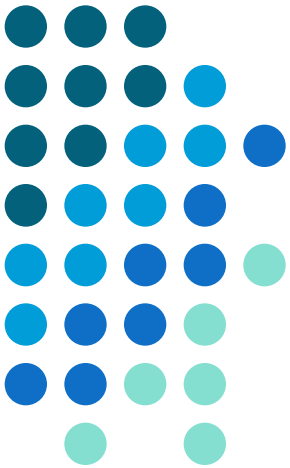
- RDBMS의 구성
  - 기본적인 데이터 저장 단위는 테이블이다.
  - 로우(ROW)와 칼럼(COLUMN)으로 구성
  - 로우는 하나의 레코드이다.
  - 칼럼은 속성을 나타낸다.

관계형 DBMS



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

2장 SQL



SQL의 개념

- SQL(Structured Query Language)란?
  - 관계 DB를 처리하기 위해 고안된 언어
  - 독자적인 문법을 갖는 DB 표준 언어
  - 데이터를 조회,입력,수정,삭제

SQL*Plus 로그인

- Command 환경에서 SQL*Plus 로그인
  - SQLPLUS *사용자계정/암호*
- 오라클 사용자 계정
  - DBA용 계정
    - 시스템 권한을 가진 사용자
  - 일반 사용자 계정

오라클 사용자 계정

계 정	암 호	설 명
SYS	DB생성 시 설정한 암호	DBA
SYSTEM	DB생성 시 설정한 암호	DBA
SCOTT	DB생성 시 설정한 암호	교육용 계정
HR	DB생성 시 설정한 암호	교육용 계정

CMD> SQLPLUS HR/HR

SQL*Plus 로그인 실패할 경우 해결 방법

- 사용자가 계정이 잠겨 있어 로그인 실패하는 경우

<에러 메시지>

the account is locked

<예>

CMD> SQLPLUS SYSTEM/123456

SQL> ALTER USER HR UNLOCK;

SQL> CONNECT HR/HR

SQL

1. SQL의 종류
    1. DDL(Data Definition Language)
    2. DML(Data Manipulation Language)
    3. TCL(Transaction Control Language)
    4. DCL(Data Control Language)

Data Definition Language(DDL)

- CREATE : 테이블 생성

<예> 부서번호, 부서명, 지역명으로 구성된 DEPT 테이블을 생성

```
SQL> CREATE TABLE DEPT(
        DEPTNO NUMBER(4),
        DNAME VARCHAR2(10),
        LOC VARCHAR2(9)
    );
```

Data Definition Language(DDL)

- DROP : 테이블을 삭제한다.

<예> DEPT02 테이블 삭제

SQL> DROP TABLE DEPT02;

- 테이블 목록 조회

SQL> SELECT * FROM TAB;

SELECT

- 테이블에 저장된 데이터를 조회

<예> DEPT 테이블의 모든 데이터를 표시

```
SQL> SELECT * FROM DEPT;
```

INSERT

- 새로운 데이터를 추가

<예> DEPT 테이블에 총무부를 추가

```
SQL> INSERT  INTO  DEPT
       VALUES(50, '총무부', '서울');
```

UPDATE

- 테이블에서 기존의 데이터를 변경

<예> DEPT 테이블에서 50번 부서의 지역명을 부산으로 변경

```
SQL> UPDATE  DEPT
       SET  LOC = '부산'
       WHERE  DEPTNO = 50;
```

DELETE

- 테이블에 저장된 데이터를 삭제

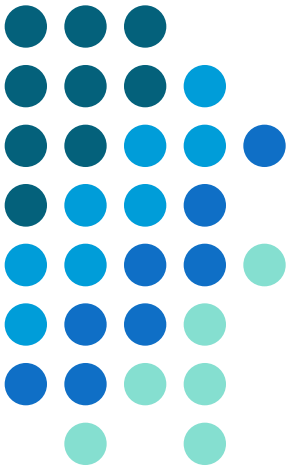
<예> DEPT 테이블에서 50번 부서를 삭제

```
SQL> DELETE  FROM  DEPT
       WHERE  DEPTNO = 50;
```

TCL(Transaction Control Language)

- COMMIT : 영구 저장
- ROLLBACK : 이전 상태로 되돌림

3장 SELECT 문



데이터를 조회하기 위한 SELECT

형식

SELECT {*, column, . . .}
FROM 테이블명;

<예> DEPT 테이블의 모든 내용 출력

```
SQL> SELECT  *
     FROM  DEPT;
```

<문제> CUSTOMER 테이블의 모든 내용 출력

칼럼 이름을 명시해서 특정 칼럼만 보기

<예> DEPT 테이블에서 부서번호와 부서명 만 출력

```
SQL> SELECT DEPTNO, DNAME
     FROM DEPT;
```

<문제> CUSTOMER 테이블에서 고객이름과 이메일, 전화번호만을 출력하는 SQL 문을 작성해보자.

- CUSTOMER 테이블의 컬럼을 확인하는 명령

```
SQL> DESC CUSTOMER;
```

WHERE 조건과 비교 연산자

두 쿼리문을 비교하자.

<예> 전체 고객을 대상

```
SQL> SELECT * FROM CUSTOMER;
```

<예> 고객 코드가 5 이상인 정보

```
SQL> SELECT * FROM CUSTOMER
        WHERE CODE >= 5;
```

WHERE 조건과 비교 연산자

연산자	의 미
=	같다.
>	보다 크다.
<	보다 작다.
>=	보다 크거나 같다.
<=	보다 작거나 같다.
<> != ^=	다르다.

<예> 고객 코드가 10 이상인 정보

```
SQL> SELECT * FROM CUSTOMER
       WHERE CODE >= 10;
```

<문제> 테이블 DEPT에서 부서코드가 30 이상인 모든 정보를 출력하라.

<문제> 고객 테이블에서 코드가 10 미만인 정보의 코드, 이름, 메일을 출력하라.

문자 데이터 조회

- 문자 데이터는 반드시 단일 따옴표 안에 표시한다.
- 대소문자를 구별한다.

<예> 이름(NAME)이 '홍길동' 인 고객

```
SQL> SELECT * FROM CUSTOMER
        WHERE NAME = '홍길동';
```

<문제> 이름이 이장미인 고객의 코드와 이름과 전화번호를 출력하라.

AND 연산자

조건1 조건2	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

<예> 부서번호가 30 이하이고 서울에 있는 부서

```
SQL> SELECT * FROM DEPT
       WHERE DEPTNO <= 30
       AND LOC = '서울';
```

<문제>코드가 5 이상 15이하인 고객

OR 연산자

조건1 조건2	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

<예> 부서번호가 10번이거나 대구에 있는 부서

```
SQL> SELECT * FROM DEPT
       WHERE DEPTNO = 10
       OR LOC = '대구';
```

<문제> 고객번호가 3이거나 5이거나 16인 고객

와일드카드(%) 사용하기

- 값을 정확히 모를 경우 사용한다.
- 몇 개의 문자가 오든 상관없다는 의미

<예> 이름이 '홍'으로 시작하는 고객

```
SELECT * FROM CUSTOMER
WHERE NAME LIKE '홍%';
```

와일드카드(%) 사용하기

<예> 이름 중에 '길'을 포함하는 고객

<예> 이름이 '동'으로 끝나는 고객

정렬을 위한 ORDER BY 절

<예>고객번호를 기준으로 오름차순으로 정렬

```
SELECT * FROM CUSTOMER
ORDER BY CODE ASC;
```

또는

```
ORDER BY CODE ;
```

내림차순 정렬을 위한 DESC

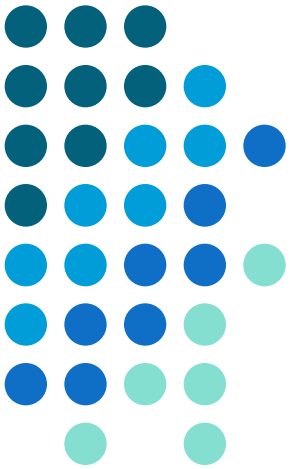
<예> 고객번호를 기준으로 내림차순으로 정렬

```
SELECT * FROM CUSTOMER
ORDER BY CODE DESC;
```

<문제> 부서정보를 부서코드가 높은 순으로 출력하라.

<문제> 고객정보를 가나다 순으로 출력하라.

4장 DML



테이블에 새로운 행을 추가하는 INSERT 문

```
CREATE TABLE DEPT01(
   DEPTNO NUMBER(2),
   DNAME VARCHAR2(14),
   LOC VARCHAR2(13)
);
```

테이블에 새로운 행을 추가하는 INSERT 문

형식

INSERT INTO *table_name*
(*column_name*, …)
VALUES(*column_value*, …);

예

- 10번 부서를 추가하자.

```
INSERT INTO DEPT01
(DEPTNO, DNAME, LOC)
VALUES(10, 'ACCOUNTING', 'NEW YORK');
```

테이블에 새로운 행을 추가하는 INSERT 문

- 칼럼명에 기술된 목록의 수보다 VALUES 다음에 나오는 괄호 안에 기술한 값의 개수가 적으면 에러가 발생한다.

```
INSERT INTO DEPT01
(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING');
```

에러 발생

테이블에 새로운 행을 추가하는 INSERT 문

- 칼럼명에 기술된 목록의 수보다 VALUES 다음에 나오는 괄호에 기술한 값의 개수가 많으면 에러가 발생한다.

```
INSERT INTO DEPT01
(DEPTNO, DNAME, LOC)
VALUES(10, 'ACCOUNTING', 'NEW YORK', 20);
```

에러 발생

테이블에 새로운 행을 추가하는 INSERT 문

- 칼럼명이 잘못 입력되었을 때에도 에러가 발생한다.

```
INSERT INTO DEPT01
(NUM, DNAME, LOC)
VALUES(10, 'ACCOUNTING', 'NEW YORK');
```

에러 발생

테이블에 새로운 행을 추가하는 INSERT 문

- 칼럼과 입력할 값의 데이터 타입이 서로 맞지 않을 경우에도 에러가 발생한다.

```
INSERT INTO DEPT01
(DEPTNO, DNAME, LOC)
VALUES(10, ACCOUNTING, 'NEW YORK');
```

에러 발생

칼럼명을 생략한 INSERT 구문

```
INSERT INTO DEPT01
VALUES (20, 'RESEARCH', 'DALLAS');
```

테이블의 내용을 수정하기 위한 UPDATE 문

형식

UPDATE *table_name*
SET *column_name1* = *value1*, *column_name2* = *value2*, …
WHERE *conditions*;

예

- 모든 부서의 위치를 대구로 수정

```
UPDATE DEPT01
SET LOC = '대구';
```

테이블의 내용을 수정하기 위한 UPDATE 문

예

- 부서번호가 10번인 정보의 위치를 서울로 수정

```
UPDATE DEPT01
SET LOC = '서울'
WHERE DEPTNO=10;
```

- 고객번호가 5인 정보의 전화번호를 053-123-1234로 수정

테이블의 내용을 수정하기 위한 UPDATE 문

예

- 김백합의 메일을 abc@aaa.com으로, 전화번호는 010-1234-5678로 수정

테이블에 불필요한 행을 삭제하기 위한 DELETE 문

형식

DELETE FROM *table_name*
WHERE *conditions*;

예

- 모든 부서를 삭제
  DELETE FROM DEPT01;
  SELECT * FROM DEPT01;

- 30번 부서를 삭제
  DELETE FROM DEPT01
  WHERE DEPTNO=30;