오라클의 SQL 문

SQL 문은 관계형 데이터베이스에서 데이터베이스나 테이블을 생성하거나 데이터의 조회, 수정, 삭제, 입력 등을 하기 위해서 사용하는 언어를 의미한다.
SQL 문은 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리하고 조회한다.
SQL 문에서 사용하는 데이터 타입 중에 null 은 매우 중요한 데이터 타입으로 데이터베이스의 null 은 값이 없는 것이 값인 값과 존재하지만 불확실한 값도 포함한다.

SQL 문의 특징:

- SQL 문은 대소문자를 구분하지 않으나 대문자 사용을 권장한다.
- SQL 문은 한 줄 또는 여러 줄에 입력할 수 있다.
- SQL 문은 여러 줄에 나누거나 단축될 수 없다.
- SQL 문에서 문은 대개 별도의 줄에 입력한다.
- SQL 문에서 가독성을 높이기 위해 들여쓰기를 사용한다.
- SQL 문에서 명령어에 대한 종료는 ;(세미콜론)으로 선언한다.
- SQL 문에서 문자 데이터와 날짜 데이터는 반드시 '(싱글 쿼터)를 사용하여 표현한다.
- SQL 문의 자체는 대소문자를 구별하지 않지만, '(싱글 쿼터)로 저장하는 리터럴은 대소문자를 구별한다.

SQL 문의 형태

데이터 정의어

DDL(Data Definition Language)은 데이터 정의어를 의미한다.
데이터 정의어는 테이블을 생성, 테이블 삭제, 테이블 재정의 등을 기능이 있다.

1. create table 문

create table 문은 테이블을 생성하기 위해 사용한다.
create table 문은 테이블을 생성할 때 맨 마지막 문에는 ,(콤마)를 생략해야 한다.

형식:

```
create table 테이블명 (
column1 datatype primary key,
column2 datatype [default 값],
column3 datatype not null,
⋮
columnN datatype
);
```

설명:

create table 테이블명:
create table 문으로 지정한 테이블명의 테이블을 생성한다.
지정한 테이블명은 중복을 허용하지 않는다.

column1 datatype primary key:
column1 은 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
primary key 는 제약조건인 기본키 제약조건을 지정하고 값은 not null 제약조건이 된다.
not null 제약조건이 설정되면 컬럼값에 null 값을 입력할 수 없다.

column2 datatype [default 값]:
column2 는 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
[default 값]은 기본값을 지정하고 날짜 입력에 주로 사용한다.

column3 datatype not null:
column3 는 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
not null 은 제약조건으로 컬럼값을 입력하지 않을 때 null 값으로 입력되는 것을 방지한다.
not null 제약조건이 설정되면 컬럼값에 null 값을 입력할 수 없다.

columnN datatype:
columnN 은 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
마지막 문이므로 ,(콤마)를 생략해야 한다.

(~); :
( )(퍼렌씨시스)로 코드블록을 적용하고 마지막에는 ;(세미콜론)을 선언해야 한다.

실습 1
사원정보를 저장하기 위한 salesman 테이블을 생성한다.

```
create table salesman (
id char(6),
name varchar2(12),
age number(3),
address varchar2(60)
);
commit;
desc salesman;
```

실습 2
게시판 정보를 저장하기 위한 board 테이블을 생성한다.

```
create table board (
num number(4),
title varchar2(30),
author varchar2(12),
content varchar2(600),
writeday date default sysdate
);
desc board;
```

2. alter table 문

alter table 문은 테이블을 수정하기 위해 사용한다.
alter table 문으로 테이블에 대한 구조를 수정할 수 있다.

형식:
alter table 테이블명 (
[add (column datatype);]
[add [constraint 제약조건명] 제약조건 (column);]
[modify (column [제약조건]);]
[drop (column);]
);

설명:
alter table 테이블명:
alter table 문으로 지정한 테이블명의 테이블 구조를 수정한다.

[add (column datatype);]:
add 문으로 ( )(퍼렌씨시스) 안의 컬럼을 추가하고 퍼렌씨시스는 생략할 수 없다.
column 은 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[add [constraint 제약조건명] 제약조건 (column);]:
add 문으로 ( )(퍼렌씨시스) 안의 컬럼에 제약조건을 추가하고 퍼렌씨시스는 생략할 수 없다.
constraint 명령어로 제약조건의 이름을 부여한다.
column 은 컬럼명을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[modify (column [제약조건]);]:
modify 문으로 ( )(퍼렌씨시스) 안의 컬럼에 지정한 데이터 타입이나 제약조건을 수정하고 퍼렌씨시스는 생략할 수 없다.
column 은 컬럼명을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[drop (column);]
drop 문으로 ( )(퍼렌씨시스) 안의 컬럼을 삭제하고 퍼렌씨시스는 생략할 수 없다.

column 은 컬럼명을 지정한다.

마지막에는 ;(세미콜론)을 선언해야 한다.

(~); :
( )(퍼렌씨시스)로 코드블록을 적용하고 마지막에는 ;(세미콜론)을 선언해야 한다.

3. add 문

add 문은 컬럼이나 제약조건을 추가하기 위해 사용한다.

형식:

alter table 테이블명 (
add (column datatype);
add [constraint 제약조건명] 제약조건 (column);
);

설명:

alter table 테이블명:

alter table 문으로 지정한 테이블명의 테이블 구조를 수정한다.

[add (column datatype);]:
add 문으로 ( )(퍼렌씨시스) 안의 컬럼을 추가하고 퍼렌씨시스는 생략할 수 없다.
column 은 컬럼명을 지정한다.
datatype 은 데이터의 타입을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[add [constraint 제약조건명] 제약조건 (column);]:
add 문으로 ( )(퍼렌씨시스) 안의 컬럼에 제약조건을 추가하고 퍼렌씨시스는 생략할 수 없다.
constraint 명령어로 제약조건의 이름을 부여한다.
column 은 컬럼명을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

(~); ;
( )(퍼렌씨시스)로 코드블록을 적용하고 마지막에는 ;(세미콜론)을 선언해야 한다.

실습

salesman 테이블에 sal 컬럼을 추가한다.

```
alter table salesman
add (sal number(7, 2));
desc salesman;
```

4. modify 문

modify 문은 테이블을 컬럼이나 제약조건을 수정하기 위해 사용한다.

형식:
alter table 테이블명 (
modify (column [제약조건]);
);

설명:
alter table 테이블명:
alter table 문으로 지정한 테이블명의 테이블 구조를 수정한다.

[modify (column [제약조건]);]:
modify 문으로 ( )(퍼렌씨시스) 안의 컬럼에 지정한 데이터 타입이나 제약조건을 수정하고
퍼렌씨시스는 생략할 수 없다.
column 은 컬럼명을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

(~); :
( )(퍼렌씨시스)로 코드블록을 적용하고 마지막에는 ;(세미콜론)을 선언해야 한다.

실습
salesman 테이블에 sal 컬럼의 최대 자릿수를 10 자리로 수정한다.

```
alter table salesman
modify (sal number(10, 2));
desc salesman;
```

5. drop 문

drop 문은 컬럼을 삭제하기 위해 사용한다.

형식:

alter table 테이블명 (
drop (column);
);

설명:

alter table 테이블명:

alter table 문으로 지정한 테이블명의 테이블 구조를 수정한다.
[drop (column);]:
drop 문으로 ( )(퍼렌씨시스) 안의 컬럼을 삭제하고 퍼렌씨시스는 생략할 수 없다.

column 은 컬럼명을 지정한다.

마지막에는 ;(세미콜론)을 선언해야 한다.
(~); :
( )(퍼렌씨시스)로 코드블록을 적용하고 마지막에는 ;(세미콜론)을 선언해야 한다.

실습

salesman 테이블에 sal 컬럼을 삭제한다.

alter table salesman
drop (sal);
desc salesman;

6. drop table 문

drop table 문은 테이블을 삭제하기 위해 사용한다.
drop table 문으로 테이블을 삭제하면 저장된 데이터도 함께 삭제된다.
외래키인 foreign key 제약조건이 설정된 경우에는 자식 테이블의 삭제가 불가능하며, 자식 테이블을 삭제하려면 먼저 부모 테이블을 삭제하거나 제약조건을 비활성 시키는 방법으로 해결할 수 있다.
foreign key 제약조건을 설정하면 부모 테이블의 기본키가 자식 테이블에서는 컬럼이 된다.
drop table 문으로 테이블을 삭제하여도 오라클 10g 버전부터는 테이블이 완전 삭제가 되지 않고 윈도우의 휴지통 기능과 같은 recyclebin 에 저장된다.
오라클 10g 버전 이후 버전부터는 삭제된 테이블에 대한 완전 삭제와 복구에 대한 명령어와 문이 만들어졌다.

형식:
[drop table 테이블명;]
[show recyclebin;]
[flashback 테이블명 to before drop;]
[purge recyclebin;]

설명:
[drop table 테이블명;]:
drop table 문으로 지정한 테이블명의 테이블을 삭제하지만, 완전 삭제가 아닌 휴지통인 recyclebin 에 저장된다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[show recyclebin;]:
show recyclebin 문으로 휴지통인 recyclebin 에 저장된 테이블의 이름과 삭제 시간 등의 정보를 확인한다.

[flashback 테이블명 to before drop;]:
flashback 테이블명 to before drop 문으로 삭제된 테이블을 복구한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

[purge recyclebin;]:
purge recyclebin 문으로 삭제된 테이블을 완전히 삭제한다.
삭제된 테이블은 flashback 문으로도 복구할 수 없다.

마지막에는 ;(세미콜론)을 선언해야 한다.

drop table 문

drop table 문은 테이블을 삭제하기 위해 사용한다.

형식 drop table 테이블명;

drop table 테이블명; :

drop table 문으로 지정한 테이블명의 테이블을 삭제하지만, 완전 삭제가 아닌 휴지통인 recyclebin 에 저장된다.

마지막에는 ;(세미콜론)을 선언해야 한다.

실습

salesman 테이블을 삭제한다.

```
drop table salesman;
select * from tab;
```

7. show 문

show recyclebin 문으로 휴지통의 정보를 확인하기 위해 사용한다.

형식 show recyclebin;

show recyclebin; :
show recyclebin 문으로 휴지통인 recyclebin 에 저장된 테이블의 이름과 삭제 시간 등의 정보를 확인한다.

실습
recyclebin 에서 삭제된 salesman 테이블의 정보를 확인한다.

show recyclebin;

8. flashback 문

flashback 테이블명 to before drop 문으로 삭제된 테이블을 복구한다.

형식 flashback 테이블명 to before drop;

flashback 테이블명 to before drop; :
flashback 테이블명 to before drop 문으로 완전히 삭제되지 않은 테이블을 복구하지만,
purge recyclebin 문으로 완전히 삭제된 테이블은 복구할 수 없다.
마지막에는 ;(세미콜론)을 선언해야 한다.

실습
삭제된 salesman 테이블을 복구한다.

```
flashback table salesman to before drop;
select * from tab;
```

9. purge 문

purge recyclebin 문으로 테이블을 완전히 삭제하기 위해 사용한다.

형식 purge recyclebin;

purge recyclebin; :
purge recyclebin 문으로 테이블을 완전히 삭제하거나 삭제된 테이블을 완전히 삭제한다.
삭제된 테이블은 flashback 테이블명 to before drop 문으로도 복구할 수 없다.
마지막에는 ;(세미콜론)을 선언해야 한다.

실습
salesman 테이블을 완전히 삭제한다.

```
drop table salesman;
purge recyclebin;
select * from tab;
```

10. rename…to 문

rename…to 문은 테이블명을 변경하기 위해 사용한다.

형식 rename 변경 전 테이블명 to 변경 후 테이블명;

rename 변경 전 테이블명 to 변경 후 테이블명; :
rename 변경 전 테이블명 to 변경 후 테이블명 문은 테이블명을 변경한다.
to 명령어는 변경할 테이블명을 지정한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

실습
board 테이블의 이름을 변경한다.

rename board to copy_board;

11. truncate table 문

truncate table 문은 테이블의 저장 공간을 삭제하기 위해 사용한다.
truncate table 문으로 삭제된 테이블은 테이블의 데이터와 저장구조가 삭제되므로 메모리를 효율적으로 관리할 수 있다.

형식 truncate table 테이블명;

truncate table 테이블명; :
truncate table 테이블명 문은 지정한 테이블명의 테이블 저장 공간을 삭제한다.
마지막에는 ;(세미콜론)을 선언해야 한다.

실습
copy_board 테이블의 저장 공간을 삭제한다.

truncate table copy_board;

12. 제약조건

1) 제약조건의 개요

제약조건(Constraints)은 컬럼에 데이터를 저장할 때 유효한 데이터 저장을 위해서 사용한다.
제약조건은 부적절한 데이터가 입력되는 것을 방지하기 위하여 사용한다.

제약조건의 특징:

- 제약조건은 테이블 레벨에서 규칙을 적용한다.
- 제약조건은 종속성이 존재할 경우 테이블 삭제를 방지한다.
- 테이블에서 컬럼값의 삽입, 갱신, 삭제될 때마다 테이블에서 규칙을 적용한다.
- 지정된 제약조건에 어긋나면 오류가 발생하며 자주 발생하는 오류는 다음과 같다.
- 컬럼값을 중복 저장할 때
  ORA-0001: 무결성 제약조건(SCOTT.CUSTOME_ID_PK)에 위배됩니다.
- 값을 저장하지 않을 때
  ORA-01400: NULL 을 ("SCOTT"."CUSTOME"."NAME") 안에 삽입할 수 없습니다.

2) 제약조건의 지정방식

(1) 컬럼 레벨 방식

형식:

```
create table 테이블명 (
column1 datatype [constraint 제약조건명] 제약조건,
column2 datatype,
column3 datatype,
⋮
columnN datatype
);
```

컬럼 레벨 방식은 컬럼을 선언하면서 제약조건도 같이 지정하는 방식이다.
not null 제약조건은 컬럼 레벨 방식으로만 사용할 수 있다.
constraint 제약조건명은 생략할 수 있다.
제약조건명은 테이블명_컬럼명_제약조건 약자로 표현한다.

(2) 테이블 레벨 방식

형식:

```
create table 테이블명 (
column1 datatype,
column2 datatype,
column3 datatype,
⋮
columnN datatype
[constraint 제약조건명] 제약조건(column1)
);
```

테이블 레벨 방식은 컬럼을 모두 지정하고 나중에 제약조건을 지정하는 방식이다.

constraint 제약조건명은 생략할 수 있다.

not null 제약조건은 사용할 수 없다.

제약조건명은 테이블명_컬럼명_제약조건 약자로 표현한다.

3) 제약조건의 형태

(1) primary key 제약조건

primary key 제약조건은 컬럼값의 중복을 방지할 목적으로 사용된다.

primary key 제약조건에서는 중복되는 컬럼값을 사용할 수 없으며 일반적으로 변경할 수 없다.

primary key 제약조건은 테이블에 대한 기본키를 생성한다.

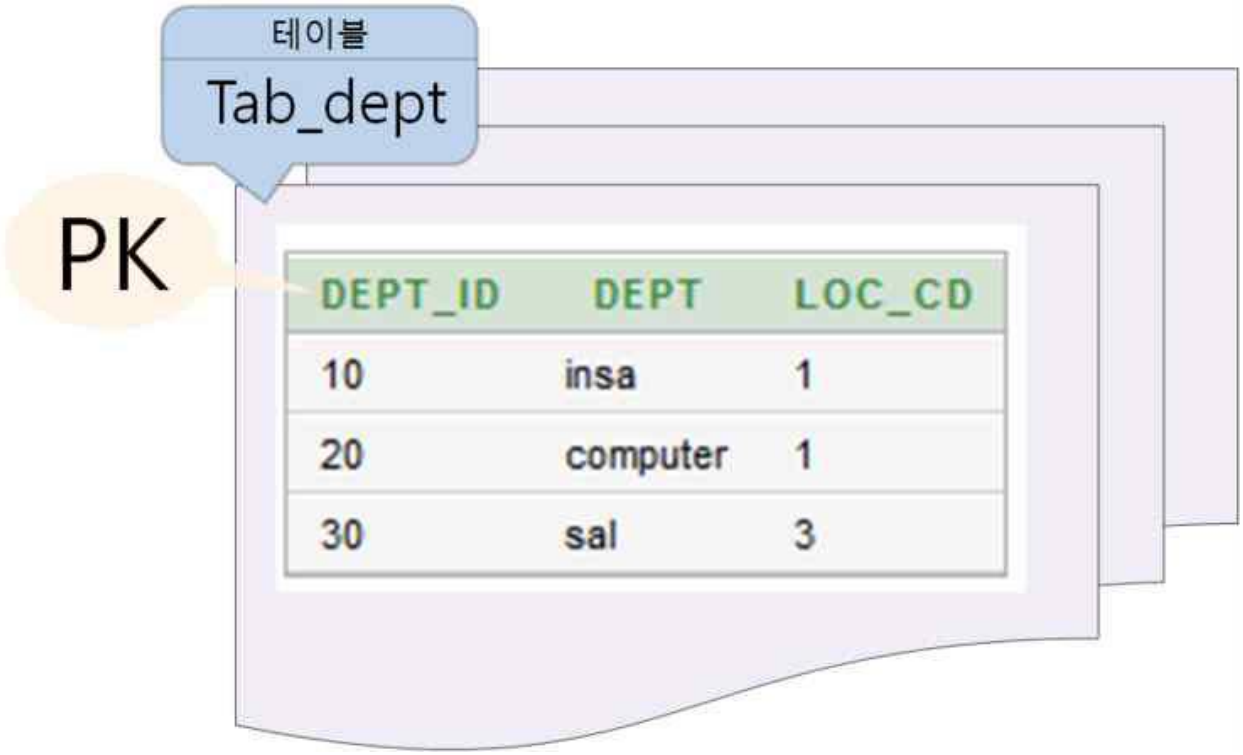
primary key 제약조건으로 생성한 기본키는 일반적으로 테이블에 하나만 생성한다.

primary key 제약조건은 테이블에서 컬럼값을 유일하게 식별하는 컬럼이다.

primary key 제약조건으로 지정한 컬럼은 null 값을 사용할 수 없다.

primary key 제약조건은 unique 제약조건과 not null 제약조건을 만족해야 한다.

primary key 제약조건의 약자는 pk 다.



(2) foreign key 제약조건

foreign key 제약조건은 테이블 간의 관계를 맺기 위해서 사용된다.

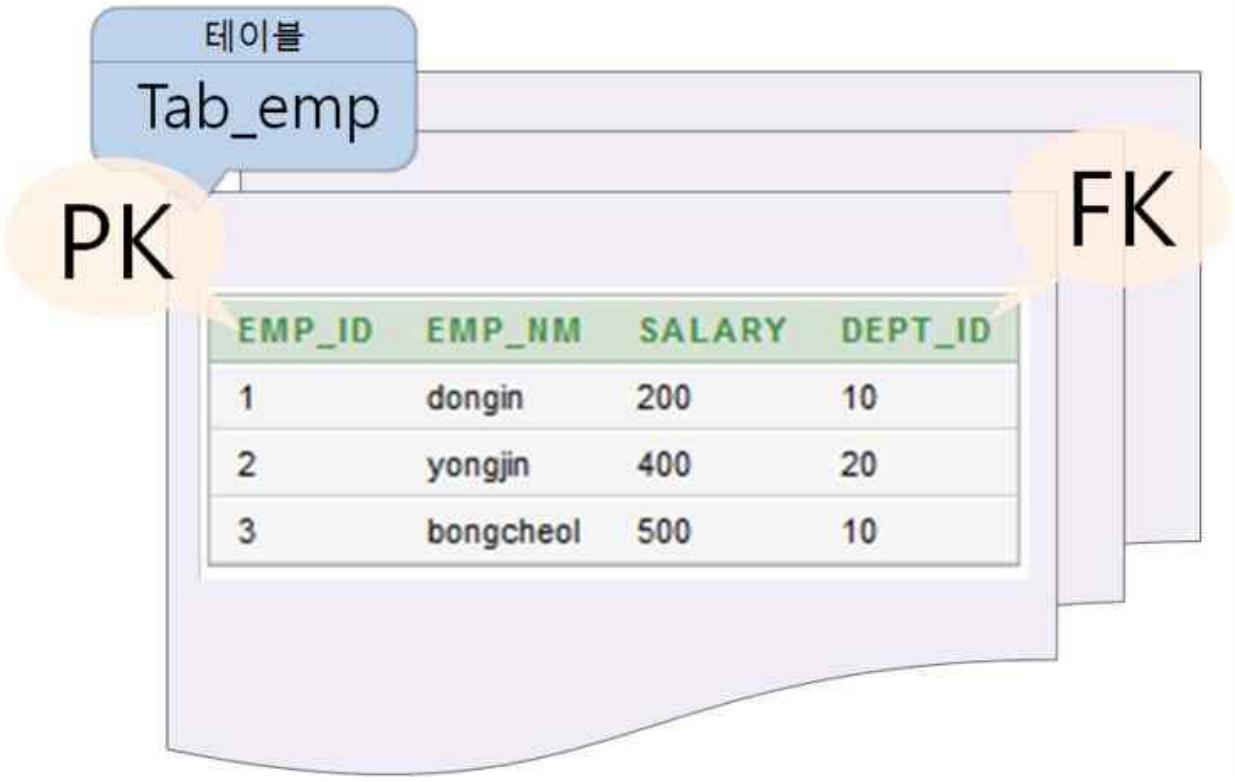
foreign key 제약조건은 테이블에 대한 외래키를 생성하며 외래키는 컬럼이다.

foreign key 제약조건으로 생성된 외래키는 부모 테이블의 기본키가 된다.

foreign key 제약조건은 다른 테이블의 primary key 제약조건을 참조해야 한다.

foreign key 제약조건을 지정한 컬럼값은 다른 테이블의 primary key 제약조건의 컬럼값이다.

foreign key 제약조건의 약자는 fk 다.



(3) unique 제약조건
unique 제약조건을 지정한 컬럼은 반드시 유일한 값을 가진다.
unique 제약조건의 약자는 uk 다.

(4) not null 제약조건
not null 제약조건을 지정한 컬럼은 반드시 값을 가져야 한다.
not null 은 제약조건으로 컬럼값을 입력하지 않을 때 null 값으로 입력되는 것을 방지한다.
not null 제약조건이 설정되면 컬럼값에 null 값을 입력할 수 없다.
not null 제약조건은 컬럼 레벨 방식으로만 사용할 수 있다.
not null 제약조건의 약자는 nn 다.

(5) check 제약조건
check 제약조건은 지정한 조건에 일치하는 컬럼값만 저장할 수 있다.
check 제약조건의 약자는 ck 다.

실습

컬럼 레벨 방식으로 제약조건을 지정하여 customer 테이블을 생성한다.

```
create table customer (
num number(4) primary key,
name varchar2(12) not null,
address varchar2(60) unique,
age number(3) check(age >= 30)
);
```

데이터 질의어

DQL(Data Query Language)은 데이터 질의어를 의미한다.
데이터 질의어는 테이블에 저장된 데이터를 조회하여 출력할 때 사용한다.
데이터 질의어의 기능:

1. 프로젝션
SQL 문에 의해 반환되는 테이블의 열을 선택한다.
필요한 수만큼 열을 선택할 수 있다.
2. 선택
SQL 문에 의해 반환되는 테이블의 행을 선택한다.
다양한 조건을 사용하여 조회되는 행을 제한할 수 있다.
3. 조인
두 테이블 사이에 링크를 지정하여 서로 다른 테이블에 저장된 데이터를 함께 가져온다.

1. select 문

select 문은 테이블에 저장된 데이터를 조회할 때 사용한다.
select 문은 전체 레코드를 조회한다.

형식 select [distinct] *|columns [as 별칭] from tables;

설명:
select 명령어
select 명령어는 표시할 컬럼이나 명령어를 지정한다.
컬럼은 하나 이상의 컬럼으로 이루어진 리스트이다.

distinct 명령어
distinct 명령어는 중복을 방지한다.

*(애스터리스크) 기호
*(애스터리스크) 기호는 모든 컬럼을 선택한다.

column 변수
column 변수는 지정된 컬럼명을 설정한다.

[as 별칭]
as 명령어는 컬럼명의 별칭이나 가상 컬럼을 생성할 수 있다.
as 명령어는 생략할 수 있다.

from tables;
from tables 문은 컬럼을 포함하는 테이블들을 지정한다.
tables 는 컬럼을 포함한 테이블들이며 ;(세미콜론)으로 종료한다.

1) 데이터의 조회

(1) 특정 컬럼의 조회

형식 select column1, column2,…columnN from table;

select 문에는 조회하고자 하는 컬럼명들을 ,(콤마)로 구분하여 지정한다.
컬럼의 호출은 select 문 뒤에 기술한 컬럼의 순서대로 호출된다.
마지막 컬럼에는 ,(콤마)를 반드시 생략해야 한다.

실습
emp 테이블에서 사원번호와 사원명을 조회한다.

select empno, ename from emp;

(2) 모든 컬럼의 조회

형식 select * from table;

컬럼명 대신에 *(애스터리스크)를 사용하여 지정한 테이블의 모든 컬럼을 조회한다.

실습
emp 테이블에서 모든 사원의 컬럼 내용을 조회한다.

select * from emp;

(3) 컬럼값의 중복 제거 조회

형식 select distinct column table;

distinct 명령어로 중복되어 출력되는 컬럼값을 제거하여 조회한다.

실습
emp 테이블에서 사원의 직급이 중복되는 값이 없도록 조회한다.

select distinct job from emp;

(4) 컬럼명의 별칭 적용

형식 select column1 [as] alias, column2 [as] alias… columnN [as] alias from table;

as 명령어를 사용하여 원하는 별칭을 지정하며 as 명령어는 생략할 수 있다.
별칭에 공백 문자나 특수문자 등을 적용하려면 "(더블 쿼터)를 사용해야 한다.
별칭의 적용은 조인문에서 컬럼명이 같을 때 유용하게 사용된다.

실습
dept 테이블에서 컬럼명 대신에 별칭을 이용해서 조회한다.

select dname as 부서명, loc as "부서 위치" from dept;

2) 데이터의 연산

(1) 산술 연산자

select 문에 산술 연산자를 적용하며 산술 연산자:

- +(플러스) : 더하기
- -(마이너스) : 빼기
- *(애스터리스크) : 곱하기
- /(슬래시) : 나누기

데이터 표시 방식을 수정하거나 계산을 수행할 때 산술식을 사용한다.
산술식은 컬럼명, 상수, 산술 연산자 등을 포함할 수 있다.
from 명령어를 제외한 SQL 문의 모든 명령어에서 산술 연산자를 사용할 수 있다.
( )(퍼렌씨시스)를 사용하여 연산자 실행 순서를 원하는 대로 지정함으로써 우선 순위규칙을 재정의할 수 있다.

실습 1
emp 테이블에서 사원의 연봉을 산술 연산자를 적용하여 조회한다.

```
select ename, sal, sal*12 from emp;
```

실습 2
emp 테이블에서 커미션을 추가한 연봉을 조회하고 null 이면 null 값을 조회한다.
특정 컬럼에 대한 컬럼값이 없는 경우 해당 값이 null 이거나 null 을 포함한다.
null 은 할당되지 않았거나 알 수 없는 값이다.
null 은 0 이나 공백과는 다르며 0 은 숫자이며 공백은 문자이다.
모든 컬럼은 null 을 포함할 수 있지만, 제약조건인 not null 이나 primary key 가 지정된 컬럼에서는 null 을 사용할 수 없다.

```
select ename, sal, sal*12, comm, sal*12+comm from emp;
```

(2) 연결 연산자

||(더블 버티컬바) 연산자로 컬럼이나 컬럼값을 다른 컬럼이나 컬럼값에 연결한다.
더블 버티컬바 연산자로 연결된 컬럼은 단일 출력 컬럼이 된다.

실습
emp 테이블에서 특정 컬럼을 더블 버티컬바 연산자로 연결하여 조회한다.

```
select empno||'-'||ename as 명단 from emp;
```

(3) select…where 문

select…where 문은 특정 조건에 맞는 레코드를 조회할 수 있다.
select…where 문을 사용하여 SQL 문에서 반환되는 컬럼값을 제한할 수 있다.
select…where 문의 조건식이 참일 때 해당 조건을 충족하는 컬럼값을 반환된다.
select…where 문의 조건식에는 컬럼값, 리터럴, 산술식, 함수, 비교 조건, 상수 등이 올 수 있다.

형식:
select [distinct] *|columns [as 별칭] from tables
where 조건식;

설명:
select 명령어
select 명령어는 표시할 컬럼이나 명령어를 지정한다.
컬럼은 하나 이상의 컬럼으로 이루어진 리스트이다.

distinct 명령어
distinct 명령어는 중복을 방지한다.

*(애스터리스크) 기호
*(애스터리스크) 기호는 모든 컬럼을 선택한다.

column 변수
column 변수는 지정된 컬럼명을 설정한다.

[as 별칭]
as 명령어는 컬럼명의 별칭이나 가상 컬럼을 생성할 수 있다.
as 명령어는 생략할 수 있다.

from tables;
from tables 문은 컬럼을 포함하는 테이블들을 지정한다.

where 조건식;
where 조건식 문은 조회할 조건을 지정하며 조건을 충족하는 컬럼값으로 SQL 문을 제한한다.
조건식은 조회할 조건에 대한 조건이나 표현식으로 ;(세미콜론)으로 종료한다.

(3) 비교 연산자

형식:
select column1, column2,…columnN from table
where column 비교 연산자 조건값;

비교 연산자는 특정 표현식을 다른 값이나 표현식과 비교하는 조건에서 사용된다.
비교 연산자:

- =(이퀄) : 같다.
- !=(엑스클러메이션 포인트 이퀄) : 같지 않다.
- >(라이트 앵글브래킷) : 크다.
- >=(라이트 앵글브래킷 이퀄) : 크거나 같다.
- <(레프트 앵글브래킷) : 작다.
- <=(레프트 앵글브래킷 이퀄) : 작거나 같다.

실습
emp 테이블에서 급여가 2000 이상인 사원만 조회한다.

```
select empno, ename, sal from emp
where sal >= 2000;
```

(4) 논리 연산자

형식:
select column1, column2, … columnN from table
where column 논리 연산자 조건값;

논리 연산자는 논리식을 구성하는 요소이다.
논리 연산자의 결과는 논리값인 참 또는 거짓을 선택하여 논리적인 계산을 수행한다.
논리 연산자 자체가 값을 반환하지는 않으며 논리 연산을 따라 최종적으로 평가된 값이 반환된다.
논리 조건은 두 구성 요소 조건의 결과를 결합하여 해당 조건을 기반으로 단일 결과를 산출하거나 단일 조건의 결과를 뒤바꾼다.

논리 연산자:

- and : 두 가지 조건이 모두 참이면 참으로 평가한다.
- or : 두 가지 조건이 모두 거짓이면 거짓으로 평가한다.
- not : 조건이 거짓이면 참으로 평가하고 조건이 참이면 거짓으로 평가한다.

실습 1
emp 테이블에서 직급이 CLERK 이고 부서번호가 10 인 사원의 정보를 조회한다.

```
select empno, ename, job, deptno from emp
where deptno = 10 and job = 'CLERK';
```

실습 2
emp 테이블에서 직급이 MANAGER 이거나 입사일이 1982 년 1 월 1 일 이후에 입사한 사원의 정보를 조회한다.

```
select empno, ename, job, hiredate from emp
where hiredate >= '1982/01/01' or job = 'MANAGER';
```

(5) between A and B 연산자

형식:
select column1, column2, … columnN from table
where column [not] between A and B;

between A and B 연산자는 특정 범위의 값을 조회할 때 사용되며 지정된 a 와 b 는 값의 범위에 포함된다.

not 명령어를 사용하면 특정 범위의 값을 조회할 때 지정된 a 와 b 는 값의 범위에서 제외한다.

실습 1

emp 테이블에서 입사일이 1981 년 5 월 5 일과 1981 년 12 월 31 일 사이의 사원정보를 조회한다.

select empno, ename, hiredate from emp
where hiredate between '1981/05/05' and '1981/12/31';

실습 2

emp 테이블에서 급여가 1500 과 4000 사이가 아닌 사원정보를 조회한다.

select empno, ename, sal from emp
where sal not between 1500 and 4000;

(6) in 연산자

형식:
select column1, column2, … columnN from table
where column [not] in (조건 리스트);

in 연산자는 여러 개의 값을 동시에 비교하여 일치하는 데이터를 얻을 때 사용한다.
in 연산자는 지정된 컬럼값의 집합에서 컬럼값을 테스트할 때 사용한다.
in 연산자를 사용하여 정의한 조건을 멤버 조건이라고도 한다.
not 명령어를 사용하면 여러 개의 값을 동시에 비교하여 일치하지 않는 데이터를 얻는다.

실습 1
emp 테이블에서 job 이 MANAGER, SALESMAN, SALESOMAN 인 사원의 정보를 조회한다.

```
select empno, ename, job from emp
where job in ('MANAGER','SALESMAN', 'SALESOMAN');
```

실습 2
emp 테이블에서 사원번호가 7369, 7521, 7698 이 아닌 사원의 정보를 조회한다.

```
select empno, ename, job from emp
where empno not in(7369, 7521, 7698);
```

(7) like 연산자

형식:
select column1, column2, … columnN from table
where column [not] like '[문자]와일드카드[문자]';

like 연산자를 사용하여 유효한 조회 문자열 값의 대체 문자 조회를 수행하며 문자 패턴을 일치시키는 작업을 대체 문자 조회라고 한다.
like 연산자는 어느 문자가 들어가도 좋다는 것을 나타내는 문자인 임의의 문자인 와일드카드와 문자열을 포함한 데이터를 검사하여 호출할 때 사용한다.
like 연산자를 사용하여 문자 패턴이 일치하는 컬럼값을 선택할 수 있다.
not 연산자를 사용하여 비교 범위를 제외한다.
not 명령어를 사용하면 유효하지 않은 조회 문자열 값의 대체 문자 조회를 수행한다.
like 연산자에 사용하는 와일드카드는 임의 리터럴 문자 결합에서 사용할 수 있다.

%(퍼센트) 와일드카드:
%(퍼센트) 와일드카드는 없거나 여러 개의 문자를 대체한다.
%(퍼센트) 와일드카드는 0 개 이상의 임의의 문자 시퀀스를 나타낸다.

_(언더스코어) 와일드카드:
_(언더스코어) 와일드카드는 하나의 문자를 대체한다.
_(언더스코어) 와일드카드는 임의의 단일 문자를 나타낸다.

※ 연산자 우선순위

–우선순위 규칙은 표현식이 평가되고 계산되는 순서를 결정한다.

–연산자 우선순위는 다음과 같다.

산술 연산자 ⟩ 연결 연산자 ⟩ 비교 조건 ⟩ is [not] null, like, [not] in ⟩

[not] between ⟩ 같지 않음 ⟩ not 논리 조건 ⟩ and 논리 조건 ⟩ or 논리 조건

실습 1

emp 테이블에서 이름이 J 문자로 시작하는 사원의 정보를 조회한다.

```
select empno, ename, hiredate, sal from emp
where ename like 'J%';
```

실습 2

emp 테이블에서 이름이 N 문자로 끝나는 사원의 정보를 조회한다.

```
select empno, ename, hiredate, sal from emp
where ename like '%N';
```

실습 3

emp 테이블에서 이름에서 두 번째 문자가 A 문자인 사원의 정보를 조회한다.

```
select empno, ename, hiredate, sal from emp
where ename like '_A%';
```

실습 4

emp 테이블에서 이름에 N 문자를 포함하는 사원의 정보를 조회한다.

```
select empno, ename, hiredate, sal from emp
where ename like '%N%';
```

실습 5

emp 테이블에서 이름에 N 문자를 포함하지 않는 사원의 정보를 조회한다.

```
select empno, ename, hiredate, sal from emp
where ename not like '%N%';
```

3) 데이터의 할당

(1) 리터럴의 할당

형식:
select column1, column2, … columnN from table
where column 비교 연산자 '문자[날짜]';

문자와 날짜 리터럴은 반드시 '(싱글 쿼터)를 사용하지만, 숫자 상수는 싱글 쿼터를 사용하지 않아도 된다.
문자 리터럴의 경우에는 대소문자를 구별하며 조회 시 대문자로 입력한 내용을 소문자로 조회하면 조회가 되지 않는다.
날짜 리터럴은 내부 날짜 형식으로 날짜를 저장하며 기본 날짜 표시 형식은 YYYY-MM-DD 이다.

실습 1
emp 테이블에서 이름이 FORD 인 사원의 정보를 조회한다.

```
select empno, ename from emp
where ename = 'FORD';
```

실습 2
emp 테이블에서 입사일이 1982 년 1 월 1 일 이후에 입사한 사원의 정보를 조회한다.

```
select empno, ename, hiredate from emp
where hiredate >= '1982/01/01';
```

(2) null 의 확인

형식:
select column1, column2, … columnN from table
where column is [not] null;

is null 문은 null 을 테스트하고 null 값의 존재 여부를 확인한다.
null 값의 의미는 다음과 같다.

- null 값은 사용할 수 없는 값이다.
- null 값은 할당되지 않은 값이다.

- null 값은 알 수 없는 값이다.
- null 값은 적용할 수 없는 값이다.

null 은 =(이퀄) 연산자를 사용하여 검사할 수 없다.

not 명령어를 사용하면 null 을 테스트하지 않고 null 값의 존재 여부를 확인하지 않는다.

실습 1

emp 테이블에서 커미션이 null 인 사원의 정보를 조회한다.

```
select empno, ename, job, comm from emp
where comm is null;
```

실습 2

emp 테이블에서 커미션이 null 이 아닌 사원의 정보를 조회한다.

```
select empno, ename, job, comm from emp
where comm is not null;
```

4) 테이블의 복사

기존에 존재하는 테이블을 이용한 새로운 테이블을 생성하며 ctas(Creating a Table from Query Results)라고 한다.
ctas 는 기존 테이블의 구조와 데이터를 그대로 복사하여 생성하지만, 제약조건은 복사가 안 되므로 ctas 로 테이블을 생성한 경우에는 제약조건을 추가하는 작업이 필요하다.

(1) 전체 테이블 복사
테이블의 구조와 데이터를 모두 복사하여 새로운 테이블을 생성한다.

형식:
create table 새로운 테이블명
as
select * from 기존 테이블명;

설명
create table 새로운 테이블명
create table 문으로 지정한 테이블명의 새로운 테이블을 생성한다.
테이블명은 중복을 허용하지 않는다.

as
as 명령어는 복사할 새로운 테이블을 지정한다.

select 명령어
select 명령어는 표시할 컬럼이나 명령어를 지정한다.
컬럼은 하나 이상의 컬럼으로 이루어진 리스트이다.

*(애스터리스크) 기호
*(애스터리스크) 기호는 모든 컬럼을 선택한다.

from 기존 테이블명
from 명령어는 컬럼을 포함하는 기존 테이블을 지정하며 ;(세미콜론)으로 종료한다.

실습

테이블의 구조와 데이터를 모두 복사하여 새로운 테이블을 생성한다.

```
create table sonboard
as
select * from copy_board;
```

(2) 테이블 구조만 복사

테이블의 데이터는 복사하지 않고 테이블의 구조만 복사한 새로운 테이블을 생성한다.

형식:
create table 새로운 테이블명
as
select * from 기존 테이블명
where 만족하지 않는 조건식;

설명:
create table 새로운 테이블명
create table 문으로 지정한 테이블명의 새로운 테이블을 생성한다.
테이블명은 중복을 허용하지 않는다.

as
as 명령어는 복사할 새로운 테이블을 지정한다.

select 명령어
select 명령어는 표시할 컬럼이나 명령어를 지정한다.
컬럼은 하나 이상의 컬럼으로 이루어진 리스트이다.

*(애스터리스크) 기호
*(애스터리스크) 기호는 모든 컬럼을 선택한다.

from 기존 테이블명
from 명령어는 컬럼을 포함하는 기존 테이블을 지정한다.

where 만족하지 않는 조건식;
where 만족하지 않는 조건식은 조회할 조건을 만족하지 않는 조건으로 지정한다.
조건식은 조회할 조건에 대한 조건이나 표현식으로 ;(세미콜론)으로 종료한다.

실습

테이블의 데이터는 복사하지 않고 테이블의 구조만 복사한 새로운 테이블을 생성한다.

```
create table board
as
select * from sonboard
where 2=1;
```

2. select...where...order by 문

select … where … order by 문은 정렬로 데이터를 조회할 수 있다.
select … where … order by 문은 특정 컬럼에 대해서 정렬한다.
정렬의 형태는 오름차순과 내림차순이 있으며 기본값은 오름차순이다.

오름차순의 정렬 방법은 다음과 같으며 내림차순은 오름차순의 반대 방법으로 정렬한다.
① 숫자 값은 가장 적은 값이 맨 앞에 표시되고 날짜 값은 가장 이른 값이 맨 앞에 표시된다.
② 문자 값은 사전 순으로 표시된다.
③ null 값은 오름차순에서는 맨 뒤에 표시되고 내림차순에서는 맨 앞에 표시된다.
④ select 리스트에 없는 열을 기준으로 정렬할 수도 있다.
수치 데이터, 문자 데이터, 날짜 데이터도 정렬이 되며 아스키코드값으로 정렬된다.

다중 정렬도 가능하며 다중 정렬은 다음과 같다.
① 하나 이상의 열로 질의 결과를 정렬할 수 있다.
② order by 문에서 열을 명시하고 컬럼명은 ,(콤마)로 구분한다.
③ 컬럼의 순서를 바꾸고자 한다면 컬럼명 뒤에 desc 명령어를 명시한다.

형식:
select [distinct] *|columns [as 별칭] from tables
where 조건식
order by column [asc/desc];

설명:
select 명령어
select 명령어는 표시할 컬럼이나 명령어를 지정한다.
컬럼은 하나 이상의 컬럼으로 이루어진 리스트이다.

distinct 명령어
distinct 명령어는 중복을 방지한다.

*(애스터리스크) 기호
*(애스터리스크) 기호는 모든 컬럼을 선택한다.

column 변수
column 변수는 지정된 컬럼명을 설정한다.

[as 별칭]
as 명령어는 컬럼명의 별칭이나 가상 컬럼을 생성할 수 있다.
as 명령어는 생략할 수 있다.

from tables;
from tables 문은 컬럼을 포함하는 테이블들을 지정한다.

where 조건식;
where 조건식 문은 조회할 조건을 지정하며 조건을 충족하는 컬럼값으로 SQL 문을 제한한다.
조건식은 조회할 조건에 대한 조건이나 표현식이다.

order by column [asc/desc]; 문
order by column [asc/desc] 문은 오름차순과 내림차순과 같은 정렬 방식을 설정한다.

asc 명령어는 오름차순으로 정렬하고 desc 명령어는 내림차순으로 정렬하며 ;(세미콜론)으로 종료한다.

1) 오름차순

asc 명령어를 컬럼명 뒤에 지정하여 오름차순으로 정렬한다.
정렬의 기본값이 오름차순이므로 asc 명령어는 생략할 수 있다.

실습
emp 테이블에서 급여가 낮은 순으로 사원 정보를 조회한다.

```
select empno, ename, sal from emp
order by sal asc;
```

2) 내림차순

desc 명령어를 컬럼명 뒤에 지정하여 내림차순으로 정렬한다.

실습 1
emp 테이블에서 입사일이 가장 최근인 순서로 사원정보를 조회한다.

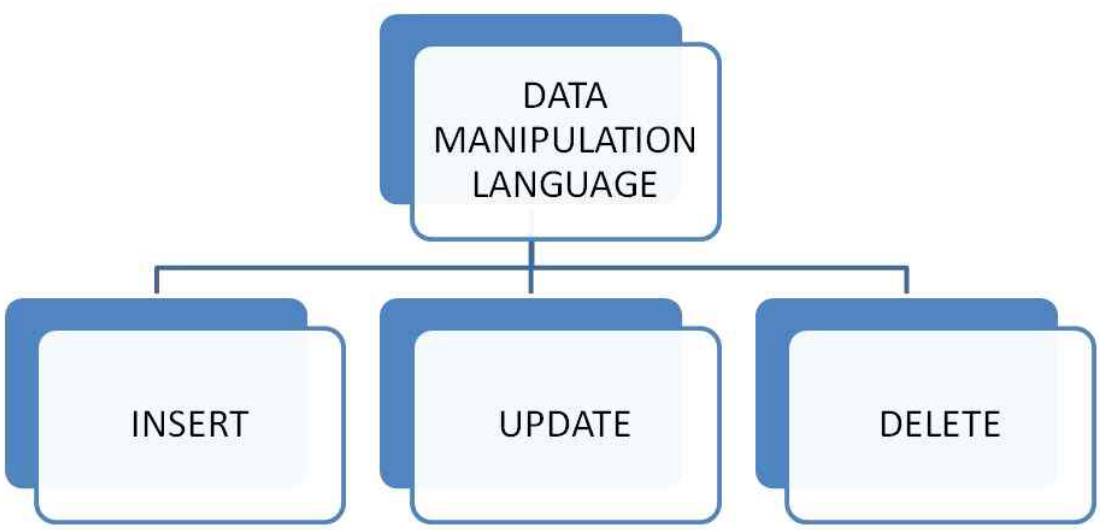
```
select empno, ename, hiredate from emp
order by hiredate desc;
```

실습 2
emp 테이블에서 급여 순으로 정렬하고 급여가 같으면 다시 이름순으로 조회한다.

```
select empno, ename, sal from emp
order by sal desc, ename;
```

데이터 조작어

DML(Data Manipulation Language)은 데이터 조작어를 의미한다.
데이터 조작어는 데이터를 조작할 때 사용한다.
데이터 조작어는 응용 프로그램이 데이터베이스에 대해 데이터 등록, 데이터 삭제, 데이터 갱신을 위한 데이터베이스 언어이다.
데이터 조작어는 관계형 데이터베이스에 대해 업데이트 등의 데이터 조작을 위해 사용된다.
데이터 조작어는 데이터베이스의 운영에 관련해 많이 사용되는 데이터 질의어와 함께 데이터를 처리한다.
데이터 조작어의 넓은 범위에서는 데이터 질의어를 포함하며 기본적인 데이터 처리 기능인 CRUD 이다.
CRUD 는 Create(생성), Read(읽기), Update(갱신), Delete(삭제)를 묶어서 일컫는 말이다.



데이터 조작어는 트랜잭션과 관련이 있으므로 SQL 문이 실행되었다고 실제 데이터베이스에 반영되지 않고 언제든지 실행 결과를 취소시킬 수 있다.

※ 데이터 조작어의 주요 오류

▫ 같은 값을 입력할 때

– ORA-00001: 무결성 제약조건(SCOTT.PK_DEPT)에 위배됩니다.

▫ 컬럼을 생략하고 모든 컬럼을 저장하지 않을 때

– ORA-00947: 값의 수가 충분하지 않습니다.

▫ 컬럼을 순서대로 값을 지정하지 않을 때

– ORA-01722: 수치가 부적합합니다.

1. insert 문

insert 문은 데이터를 입력한다.
insert 문은 테이블에 새로운 데이터를 입력할 때 사용한다.

형식:
insert into 테이블명(column1, column2, … columnN)
values(값 1, 값 2, … 값 N);
insert into 테이블명(column1, column2, … columnN)

설명
insert into 테이블명(column1, column2, … columnN) 문은 지정한 테이블명으로 테이블과 데이터를 저장할 컬럼을 지정한다.
컬럼은 생략할 수 있지만, 명시적으로 컬럼을 지정하는 것을 권장한다.
컬럼을 생략하는 경우에는 반드시 테이블의 모든 컬럼에 데이터를 저장한다.

values(값 1, 값 2, … 값 N);
values 문에는 컬럼과 일대일 대응이 되도록 값들을 지정하고 ;(세미콜론)으로 종료한다.
테이블의 컬럼 순서대로 값을 지정해야 한다.
컬럼값을 명시하지 않으면 null 값이 자동으로 저장된다.

실습 1
board 테이블에 새로운 컬럼값을 저장한다.

```
insert into board(num, title, author, content)
values(1, '테스트', '홍길동', '테스트입니다.');
select * from board;
```

실습 2
dept 테이블에 새로운 부서 정보를 저장한다.

```
insert into dept(deptno, dname, loc)
values(50, '개발부', '서울');
select * from dept;
```

실습 3
부모 키가 존재하지 않는 데이터를 입력하면 오류가 발생한다.

```
insert into emp(empno, ename, deptno)
values(6789, '홍길동', 70);
```

2. update 문

update 문은 데이터를 수정한다.
update 문은 테이블에 저장된 데이터를 수정하기 위해서 사용한다.

형식:
update 테이블명
set column = 새로운 값
where 조건식;

설명:
update 테이블명
update 테이블명 문은 지정한 테이블명으로 수정할 테이블을 지정한다.

set column = 새로운 값
set column = 새로운 값 문은 column 에 새로운 값을 입력한다.

where 조건식;
where 조건식 문은 수정할 조건을 지정하며 조건을 충족하는 컬럼값으로 SQL 문을 제한한다.
where 문을 생략하는 경우에는 지정한 컬럼의 모든 컬럼값이 수정되므로 주의해야 한다.
where 문을 지정하면 조건식과 일치하는 컬럼값만 수정된다.
조건식은 수정할 조건에 대한 조건이나 표현식으로 ;(세미콜론)으로 종료한다.

실습
dept 테이블의 부서번호가 50 인 부서명을 기획실로 변경한다.

```
update dept
set dname = '기획실'
where deptno = 50;
select * from dept;
```

3. delete 문

delete 문은 데이터를 삭제한다.
delete 문은 테이블에 저장된 데이터를 삭제하기 위해서 사용한다.

형식:
delete from 테이블명
where 조건식;

설명:
delete from 테이블명
delete from 테이블명 문은 지정한 테이블명으로 삭제할 테이블을 지정한다.

where 조건식;
where 조건식 문은 삭제할 조건을 지정하며 조건을 충족하는 컬럼값으로 SQL 문을 제한한다.
where 문으로 조건을 주어 조건에 해당하는 컬럼만 삭제할 수 있다.
where 문을 생략하는 경우에는 지정한 컬럼의 모든 컬럼값이 삭제되므로 주의해야 한다.
where 문을 지정하면 조건식과 일치하는 컬럼값만 삭제된다.
조건식은 삭제할 조건에 대한 조건이나 표현식으로 ;(세미콜론)으로 종료한다.

실습 1
dept 테이블의 사원번호가 50 인 컬럼값을 삭제한다.

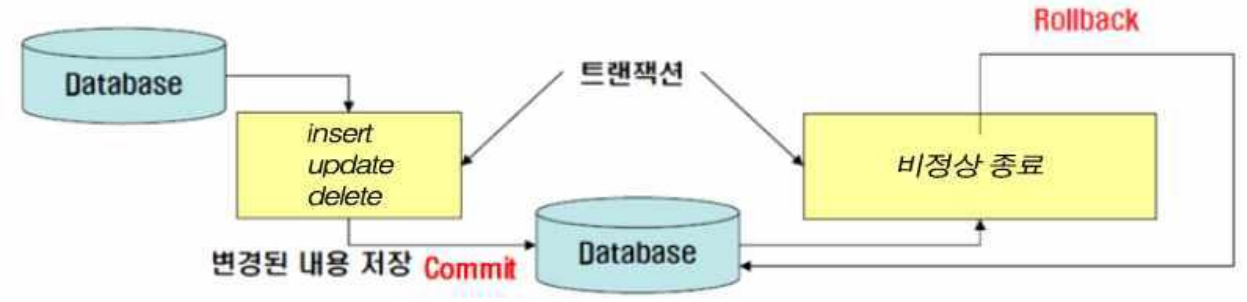
```
delete from dept
where deptno = 50;
select * from dept;
```

실습 2
외래키가 존재하는 부모 테이블을 삭제하면 오류가 발생한다.

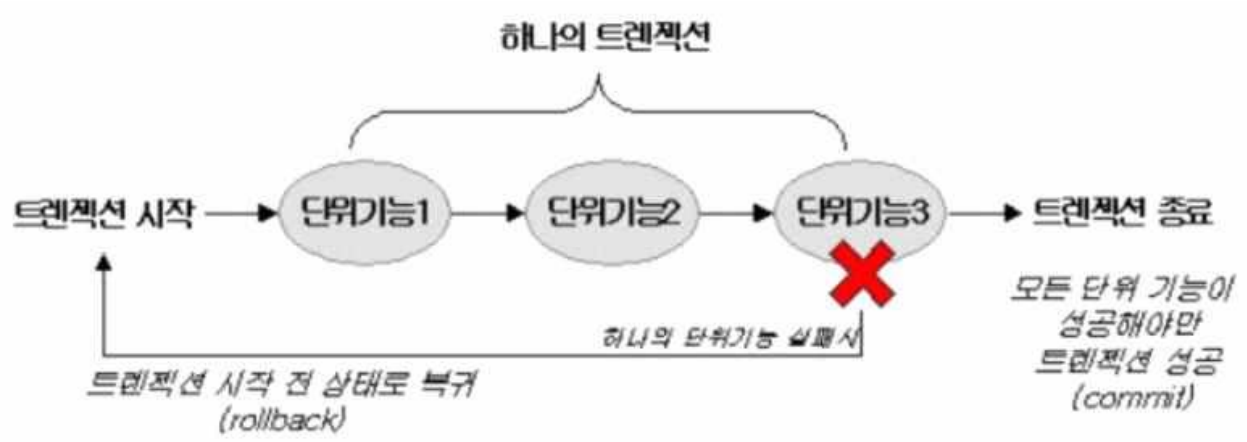
```
delete from dept
where deptno = 30;
```

데이터 처리어

TCL(Transaction Control Language)은 데이터 처리어를 의미한다.
데이터 처리어는 데이터의 작업처리인 트랜잭션에 적용한다.
트랜잭션(transaction)은 데이터베이스 시스템에서 사용되는 작업처리의 최소 단위이다.
트랜잭션은 데이터베이스 작업처리를 모두 일관되게 하려고 존재한다.
트랜잭션은 데이터 일관성을 제공하므로 만약 두 가지의 데이터베이스에 명령을 주었을 때 첫 번째 명령의 처리는 올바르게 되었지만 두 번째 명령의 처리가 올바르지 못하다면 일관성이 유지되지 않으므로 두 가지 명령 중 하나라도 실패하면 원점으로 되돌린다.
하나에서 여러 개의 데이터 조작어를 실행해야 한다면 반드시 트랜잭션 처리가 되어야 한다.



트랜잭션은 여러 개의 데이터 조작어가 하나의 논리적인 실행 단위로 묶어서 단위로 묶인 데이터 조작어 중에 하나라도 정상적으로 수행되지 않고 실패하면 전체를 취소하는 기능이다.
트랜잭션 이용하면 하나의 트랜잭션으로 묶인 작업을 전부 실행되든지 전부 취소되게 처리할 수 있다.



데이터베이스 작업은 초기 상태에서 프로세스에 의해서 트랜잭션으로 각 데이터베이스 작업을 처리하게 된다.

데이터베이스 작업이 오류 없이 모두 잘 작동된다면 커밋을 수행하고 트랜잭션 처리가 마무리되지만, 여러 데이터베이스 작업 중 하나의 문제라도 발생하게 되면 롤백으로 돌아가고 초기 상태 이후에 처리한 데이터베이스 작업은 모두 무효 처리된다.
프로세스 단계에서 처리되는 작업은 커밋이 실행되지 않는 한 데이터베이스에 업데이트 작업이 이루어지지 않는다.

트랜잭션의 명령에는 크게 처리한 작업을 모두 반영하도록 하는 커밋(commit)과 처리한 작업을 모두 되돌리는 롤백(rollback)이 있다.

묵시적인 트랜잭션은 비정상적인 방법으로 종료되면 자동으로 롤백으로 트랜잭션이 종료되고 정상적인 방법으로 종료하면 커밋으로 트랜잭션이 종료된다.

1. commit 문

형식 commit;

commit 문은 트랜잭션을 저장하며 ;(세미콜론)으로 종료한다.
commit 문으로 트랜잭션 작업을 완료 시키면 실제적인 물리적 디스크에 저장한다.
외부 응용 프로그램에서 데이터를 입력할 때 commit 문으로 저장하지 않으면 실제 데이터가 저장되지 않는다.

실습
입력 작업처리를 완료한다.

```
insert into dept(deptno, dname, loc)
values(50, '인사과', '부산');
commit;
select * from dept;
```

2. rollback 문

형식 rollback;

rollback 문은 트랜잭션을 취소한다.
rollback 문으로 트랜잭션 작업을 취소시키면 실제적인 물리적 디스크에서 정보를 취소한다.

실습
삭제 작업처리를 취소한다.

```
delete from emp;
select * from emp;
rollback;
select * from emp;
```

3. savepoint 문

형식 savepoint 세이브포인트명;

savepoint 문은 트랜잭션 내의 책갈피 기능을 하고 ;(세미콜론)으로 종료한다.
savepoint 문으로 현재 트랜잭션 내에 저장 지점을 만든다.

실습
저장 지점만 작업처리를 취소한다.

```
insert into dept(deptno, dname, loc) values(60, '총무과', '광주');
savepoint a;
insert into dept(deptno, dname, loc) values(70, '개발과', '대구');
savepoint b;
select * from dept;
delete from dept where deptno = 60 or deptno = 70;
select * from dept;
rollback to savepoint a;
select * from dept;
```

데이터 제어어

DCL(Data Control Language)은 데이터 제어어를 의미한다.
데이터 제어어는 최고관리자 계정인 sys 계정에서 권한을 부여하고 삭제하는 역할을 하면 DBA 가
담당한다.
데이터베이스를 접속하기 위해서는 계정과 암호가 필요하다.

계정에 접속한 후에 테이블을 생성하고 데이터를 조회, 입력, 수정, 삭제 등을 하기 위해서는 특별한 권한이 필요하다.
권한의 종류도 시스템 권한과 객체 권한으로 나누어진다.
여러 개의 권한을 묶어서 롤(role)이라는 개념으로 사용한다.
개별적인 권한보다 롤을 사용하는 것이 더 효율적이며 새로운 계정과 권한 부여는 sys 계정에서만 가능하다.

1. 계정의 관리

1) 계정 생성

최고관리자 sys 계정에서 create user 문으로 새로운 계정을 생성한다.

형식 create user 계정 identified by 비밀번호;

설명:
create user 계정
create user 문으로 계정을 생성한다.

identified by 비밀번호;
identified by 문으로 비밀번호 생성하고 ;(세미콜론)으로 종료한다.

실습
최고관리자 SYS 계정으로 접속하여 test 계정과 생성한 test 계정의 비밀번호를 생성한다.

create user test identified by 12345;

2) 계정 변경

최고관리자 sys 계정에서 alter user 문으로 계정을 변경한다.

형식 alter user 계정 identified by 새로운 비밀번호;

설명:
alter user 계정
alter user 문으로 계정을 변경한다.

identified by 새로운 비밀번호;
identified by 문으로 기존 비밀번호를 변경하고 ;(세미콜론)으로 종료한다.

실습
최고관리자 SYS 계정으로 접속하여 test 계정의 비밀번호를 변경한다.

alter user test identified by 1234;

3) 계정 삭제

최고관리자 sys 계정에서 drop user 문으로 계정을 삭제한다.

형식 drop user 계정 cascade;

설명:
drop user 계정
drop user 문으로 계정을 삭제한다.

cascade 명령어
cascade 명령어로 기존 계정을 완전히 삭제하고 ;(세미콜론)으로 종료한다.

실습
최고관리자 SYS 계정으로 접속하여 test 계정을 삭제한다.

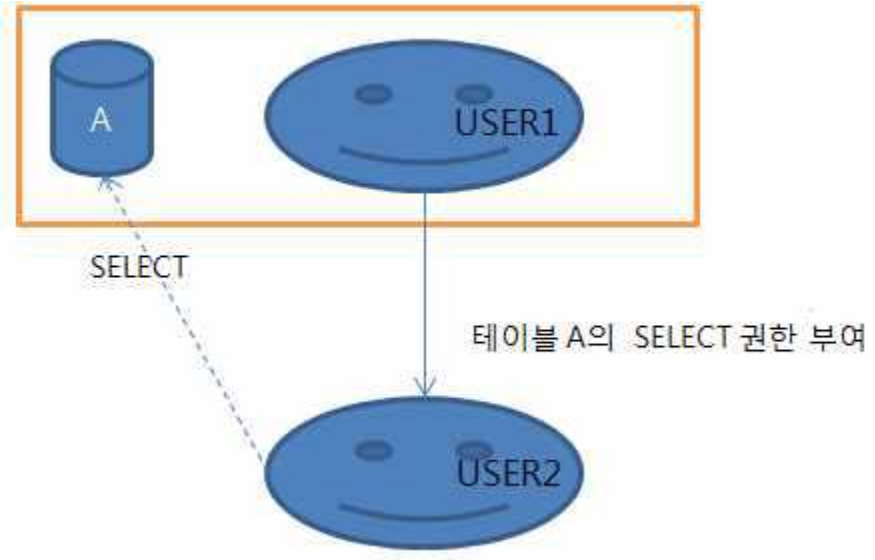
drop user test cascade;

2. 계정의 권한

1) 권한 부여

형식 grant create 권한 및 롤 to 계정;

최고관리자 sys 계정에서 grant create 문으로 계정에 권한을 부여한다.
grant create 문으로 권한과 롤을 설정하고 to 명령어로 계정에 부여한다.



설명:
grant create 권한 및 롤
grant create 문으로 권한 및 롤을 부여한다.
롤은 사용자에게 허가할 수 있는 권한들의 집합이다.

to 계정;
to 명령어로 권한을 부여할 계정을 지정하고 ;(세미콜론)으로 종료한다.

실습 1
최고관리자 SYS 계정으로 접속하여 test 계정에 접속 권한을 부여한다.

grant create session to test;

실습 2
최고관리자 SYS 계정으로 접속하여 test 계정에 테이블 관련 권한을 부여한다.
grant create table to test;

실습 3
최고관리자 SYS 계정으로 접속하여 test 계정에 시퀀스 권한을 부여한다.

grant create sequence to test;

실습 4
최고관리자 SYS 계정으로 접속하여 test 계정에 뷰 생성 권한을 부여한다.

grant create view to test;

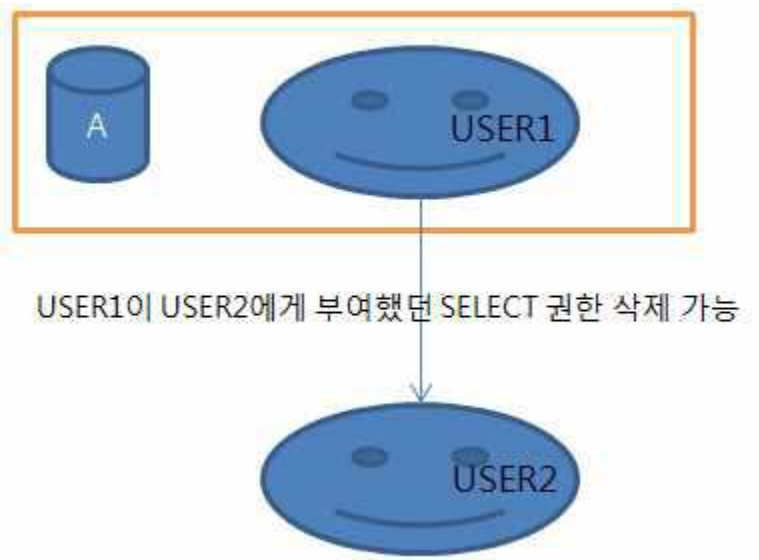
실습 5
최고관리자 SYS 계정으로 접속하여 test 계정에 접속과 자원에 관한 롤을 부여한다.

grant connect, resource to test;

2) 권한 삭제

형식 revoke create 권한 및 롤 from 계정;

최고관리자 sys 계정에서 revoke create 문으로 계정에 권한을 삭제한다.
revoke create 문으로 권한과 롤을 설정하고 from 명령어로 계정에 부여한다.



설명:
revoke create 권한 및 롤
revoke create 문으로 권한 및 롤을 삭제한다.
롤은 사용자에게 허가할 수 있는 권한들의 집합이다.

from 계정;
from 명령어로 권한을 삭제할 계정을 지정하고 ;(세미콜론)으로 종료한다.

실습
최고관리자 SYS 계정으로 접속하여 시퀀스 권한을 삭제한다.

revoke create sequence from test;