1. 오라클에서 **테이블 구조를 보는 명령어**

오라클에서 **desc(describe) 테이블명;**

2. **오라클 자료형**

가.**number(10)** 라고 지정하면 **정수형숫자 10자리까지 저장**한다.

나.**number(10,2) 라고 지정하면 실수형 자료형**이다. 즉 **소수점 둘째짜리 까지 포함한 총자리수 10자리 실수형 숫자값을 저장**한다.

다.**Date 형**은 **날짜형 자료형**으로서 **오라클 함수에서 sysdate 날짜값을 반환하는 함수**와 주로 같이 사용한다. **mysql 에서는 날짜/시간값을 반환하는 함수로 now() 함수를 사용한다.**

라.**char형**은 **고정 문자열 자료형 -> 기억장소 낭비의 단점**이 있다.

마.**varchar2 형**은 **가변형 문자열 자료형**.-> **기억장소 절약의 장점**이 있다.

3. 오라클에서 **동일한 중복 레코드를 한번만 출력**

형식) **select distinct 필드명 from 테이블명;** 즉 **distinct 예약어**를 사용한다.

4. **SQL plus 편집** 명령어

1. **LIST(L)** : 버퍼의 내용을 나타내기 위한 명령어이다. 약자로 L로 사용해도 된다. 오라클에서는 바로 이전에 실행했던 쿼리문만 버퍼에 저장된다. 이 버퍼에 저장된 쿼리문을 출력하고 싶으면 List를 사용한다.

2. **/** : 이전에 버퍼에 저장된 쿼리문을 출력하지 않고 바로 실행한다.

3. **run(R)** : 버퍼에 저장된 쿼리문을 출력하고 실행한다. 이 명령어는 List+ / 랑 같다.

1. SQL PLUS 파일 명령어

1. EDIT(ED) : 파일의 내용을 유닉스의 VI 나 윈도우의 NOTEPAD 와 같은 에디터를 읽어 편집할 수 있게 해준다.

EDIT(ED) 파일명

실행했던 **쿼리문 마지막 세미클론(;)이 편집기에서는 /로 대체된다는 점에 주의**한다.

저장되는 파일의 확장자는 *.buf 형식이 된다.

2. host: **오라클을 종료하지 않고 도스 프롬프트로 빠져나갈때 사용한다. 다시 sql 프롬프트 돌아올려면 exit** 을 사용한다.

3. save : ***.sql 파일로 쿼리문을 저장**하고자 할때 사용한다. **파일이름만 기술하고 확장자를 생략하면 기본확장자 sql 로 지정**된다. save 옵션으로 replace 또는 append 를 사용할 수 있다. **이미 존재하는 파일에 새로운 내용으로 덮어쓰길 원하면 replace 옵션을 사용** 한다. **append 옵션을 사용하면 이미 존재하는 파일 끝에 마지막으로 실행한 명령어를 추가한다.**

4. **@파일명** : **파일명에 저장된 쿼리문을 실행할때 사용**한다. **확장자를 생략하면 .sql** 로 인식한다.

5. SPOOL : 수업시간에 학습한 쿼리문은 물론이거니와 실행 결과까지도 파일에 저장해 두 었다가 복습을 하거나 서브 노트를 만들고 싶을 경우 주로 사용한다.

SAVE 명령어는 SQL 문 자체를 저장하는 명령어이며, **spooL 명령어는 sql 문과 실행된 쿼리문 결과를 파일로 기록하는 명령어**이다. 즉 , **화면에 보이는 내용 전체를 갈무리해서 하나의 파일**로 만든다.

형식) SPOOL 파일이름

기본 확장자는 ***.LST** 이다. **spool 해제를 위한 명령어로 SPOOL OFF** 를 사용한다.

SPOOL OFF 명령어는 SPOOL 화면 갈무리 작업을 중단한다. **해제하기 전 까지의 여러 쿼리문을 모두 저장**한다. **SPOOL 명령어를 사용 할때 화면을 갈무리한 내용을 저장하기 위해서는 반드시 SPOOL OFF** 를 해 주어야 한다. **만약 SPOOL OFF 를 하지 않고 오라클을 종료 해 버리면 지금까지 갈무리한 내용이 저장되지 않는다.**

6. get : **save 명령어를 사용하여 저장한 sql 명령어를 다시 가져오고자 할 경우 이때 사용하는 명령어가 get** 이다. **확장자를 기술하지 않으면 기본확장자로 .sql 로 인식**한다.

형식)get 파일명

1. **LIKE 연산자와 와일드 카드 문자**

1. **LIKE 연산자**는 **검색하려는 값을 정확히 모를 경우에도 검색할 수 있도록 와일드 카드와 함께 사용하여 원하는 내용을 검색**하게 한다.

형식) **컬럼명 LIKE 형태**;

2. **와일드 카드**
   가. **%: 하나 이상의 모르는 임의의 문자와 매핑 대응**한다.
   나. _: **단 하나 모르는 문자와 매핑 대응**한다.

2. **NULL의 의미**

1. **NULL**은 **미확정, 알 수 없는 값**을 의미

2. NULL값을 **= 연산자로 판단할 수 없다.**

3. **특정 필드 값이 NULL 값인지를 비교할 경우**에는 **IS 연산자를 사용 즉 IS NULL로 판단**

4. 특**정 필드 값이 NULL이 아닌 자료만 추출**하기 위해서는 **IS NOT NULL 연산자**를 사용한다.

3. **숫자 함수**

1. **ABS함수: 절대값을** 구하는 함수이다. 주어진 값이 음수인 경우 양수로 표현된다.

2. **FLOOR함수: 소수점 아래를 버리는 함수**이다.

3. **ROUND함수: 지정한 자릿수 이하에서 반올림한 결과를 구하는 함수**이다.
   형식) ROUND(대상, 자릿수)

예를 들면

ROUND(34.5678, 2) 하면 두번째 인자값이 2이면 소수점 이하 세번째 자리에서 반올림하여 소수점 이하 두번째 자리까지 표시된다.

4. TRUNC함수: 지정한 **자릿수 이하를 버린 결과를 구하는 함수**이다.

예를 들면

TRUNC(34.5678, 2) 하면 **두번째 인자값이 2이면 소수점 이하 3번째 자리에서 버림연산을 한 후 소수점 이하 2번째 자리까지 표시**된다. 결국 **34.56**이 구해진다.

5. MOD함수 : 나눗셈을 한 후 몫이 아닌 **나머지를 구하는 함수**이다.

1. DECODE 함수 특징

1. DECODE 함수는 프로그램 언어에서 가장 많이 사용하는
switch case 문과 같은 기능을 한다. 즉 여러 가지 경우에 선택할 수 있게 한다.

사용하는 형식)

```
DECODE (표현식,조건1,결과1,
                조건2,결과2,
                조건3,결과3,
                기본결과n
        )
```

2. CASE 함수

1. CASE 함수 역시 **여러가지 경우에서 하나를 선택하는 함수**이다.

2.CASE 함수는 프로그램 언어의 if else if else
다중 조건문과 같은 기능을 한다.

형식)

```
CASE WHEN 조건1 THEN 결과1
     WHEN 조건2 THEN 결과2
     WHEN 조건3 THEN 결과3
     ELSE 결과N
END
```

1. **그룹함수** 종류

1. **SUM(합계), AVG(평균), COUNT(총레코드 개수),**

   MAX(최대값), MIN(최소값), STDDEV(표준편차),

   VARIANCE(분산)

2. **그룹함수를 사용하면 comm 필드 보너스 총합을 구해도 다른 연산자와 달리 null을 제외하고 총합을 구한다. 즉 그룹함수는 null값을 제외한다.**

3. **count함수는 null값에 대한 개수를 세지 않는다.**

2. **group by 문**

1. 어떤 특정 컬럼 값을 기준으로 그룹함수를 사용하고자 할 경우 select 문 다음에 group by 문을 추가하고 해당 컬럼을 기술한다.
   예)**부서별 급여 평균을 구하는 경우** 사용.

3. **where 조건식과 having 조건식 비교**

1.**where** 조건식
   가.where 절은 테이블에서 데이터를 가져올 때 **특정 조건에 맞는 자료만을 검색**할 때 사용한다.

2.having 조건식
   가.**그룹함수 사용 후 나온 결과값 중에서 원하는 조건에 맞는 자료만 추출**하고자 할때 사용한다.

   나.**having 문에는 그룹함수를 적용한 컬럼이 조건식으로 온다.**

1. 조인이란?

가. 하나이상의 테이블을 서로 합쳐서 데이터를 조회하기 위해서 사용되는 것을 조인(join)이라 한다.

2. cross join이란?

가. 2개 이상의 테이블이 조인될 때 where절에 의한 공통되는 컬럼에 의한 결합이 발생하지 않는 경우를 뜻한다.

나 .cross join을 하면 테이블 전체 행에 대한 컬럼이 조인된다.

3. Equi JOIN 특징

1. Equi Join은 가장 많이 사용하는 조인 방법으로 조인 대상이 되는 두 테이블에서 공통적으로 존재하는 컬럼의 값이 일치되는 행을 연결하여 결과를 생성하는 조인 기법이다.

2.두 테이블을 조인하려면 일치되는 공통 컬럼을 사용해야 한다. 컬럼의 이름이 같으면 혼동이 오기 때문에 컬럼 이름앞에 테이블명을 명시해야 한다.

4 . Non Equi Join 특징

1. 조인 조건의 특정 범위 내에 있는지를 조사하기 위해 where 조인 조건으로 =연산자 이외의 비교 연산자를 사용한다.

2. where 절의 특정 범위를 급여등급으로 하는
SALGRADE테이블을 생성한다.

grade필드(급여등급)	LOSAL(최소급여)	HISAL(최대급여)
1	700	1200
2	1201	1400
3	1401	2000

3. WHERE 조건절에서 특정범위내의 조인 조건으로
검색하는 기법을 NON EQUI JOIN이라 한다.

미국 표준협회(ANSI)제시한 Join문

1. ANSI Cross Join문

가. ANSI Cross Join문은 From절 다음에 쉼표없이 cross join 이라는 명확한 단어를 사용해서 조인을 하는 방법이다.

2. ANSI Inner join문

1. Ansi Inner join문은 From절 다음에 Inner Join 이라는 명확한 단어를 사용하여 조인할 테이블 명을 명시하고 on절문에 조인 조건을 명시하는 조인 방법이다.

2. 이 조인문은 두 테이블의 동일 컬럼을 기준으로 조인한다.

3. using 문을 사용한 조인 조건
가. 조인을 정의한 컬럼명이 두 테이블에서 모두 동일하다면 using 절에서 조인할 컬럼을 지정하여 보다 더 간단히 조인을 할수 있다.

나. using 절문에는 테이블명 또는 테이블 명 별칭을 사용할 수 없다.

3. Natural Join문

1. 조건절을 생략하고 Natural Join을 사용하면 자동적으로 모든 컬럼을 대상으로 공통 컬럼을 조사하여 내부 조인을 실행한다.

1. ANSI Outer Join 기법

1. Outer Join은 어느 한쪽 테이블에는 해당하는 데이터가 존재하는데 다른쪽 테이블에는 데이터가 존재하지 않는 경우 그 데이터가 출력되지 않는 문제점을 해결하기 위해 사용하는 조인기법이다.

2.ANSI Outer Join의 종류:
LEFT Outer join, Right Outer Join,
Full Outer join 이 있다.

가. **LEFT Outer Join**기법 : From 절다음에 테이블명을 기술할때 왼쪽,오른쪽에 기술하는데 **오른쪽 테이블 명에 데이터가 없는 경우 사용하는 조인기법**이다.

나. **Right Outer Join기법** : From 절 다음에 테이블명을 기술할때 왼쪽,오른쪽에 기술하는데 **왼쪽 테이블에 데이터가 없는 경우 사용하는 조인기법**이다.

다. Full Outer Join기법: **Left Outer Join과 Right Outer Join기법을 합한 형태** 이다.

3. Outer Join도 inner join처럼 두 테이블 간의 조인 조건에 사용되는 컬럼명이 같다면 On문 대신 Using절문을 사용할 수 있다.물론 이런 경우도 테이블명.컬럼명 형태가 아닌 컬럼명만 명시한다.

1.서브쿼리문

가. **서브 쿼리문 역시 특정 테이블에서 검색 한 결과를 다른 테이블에 전달하여 새로운 결과를 검색 하는 경우에 사용**한다.

2. 서브 쿼리문 구조

```
select dname from dept
    where deptno=( select deptno
                   from emp
                   where ename='scott');
```

가. **서브쿼리**는 **하나의 select문장의 절안에 포함된 또 하나의 select문 절**이다. 그렇기 때문에 **서브 쿼리를 포함하고 있는 쿼리문을 메인 쿼리**라 하고, **포함된 또 하나의 쿼리를 서브 쿼리**라 한다.

나.**서브 쿼리문은 비교 연산자의 오른쪽에 기술해야 하고 반드시 소괄호 안에 넣어야 한다.** **서브 쿼리는 메인 쿼리가 실행되기 전 한번만 실행**된다.

다. 서브 쿼리는 **단일행 서브 쿼리와 다중행 서브 쿼리**가 있다.

3. 단일 행 서브 쿼리문

가. **단일 행 서브쿼리는 수행 결과가 오직 하나의 행만 반환하는 서브 쿼리**를 뜻한다.

나.단일 행 서브쿼리는 오직 하나의 행으로 반환된 결과값을 메인 쿼리로 보내는 데 **매인 쿼리의 where**

절에는 단일행 비교연산자를 사용한다.

다. 단일행 비교연산자 종류
=(같다), >, >=, <, <=, <>(같지 않다.)

1. 다중행 서브쿼리

1.다중행 서브쿼리: 서브 쿼리에서 반환되는 결과가 하나 이상의 행일때 사용하는 서브 쿼리를 말한다.

2. 다중행 서브 쿼리는 단일행 서브쿼리 연산자(=,>, >=,<,<=,<>)를 사용할 수 없고,다중행 연산자를 사용해야 한다.

3. 다중행 서브쿼리 연산자 종류

가. IN연산자 : 메인 쿼리의 비교 조건에서 서브 쿼리의 출력 결과와 하나라도 일치하면 메인 쿼리의 WHERE 절이 참이 되는 연산자를 뜻함.

나. >ALL 연산자 : 서브 쿼리의 모든 결과값중에서 최대값보다 크면 참이되는 연산을 뜻함.

다. > ANY 연산 : 서브 쿼리의 모든 결과값중에서 최소값보다 더 크면 참이 되는 연산을 뜻함.

1. alter table문

1. alter table문: 기존 테이블 구조를 변경하기 위한 쿼리문이다.

2. 기존 테이블에 새로운 컬럼을 추가하기

```
alter table 테이블명
add(추가할 컬럼명 자료형(크기));
```

3. 기존 컬럼 속성 변경

```
alter table 테이블명
modify(필드명 자료형(크기));
```

4. 기존 컬럼 삭제

```
alter table 테이블명
drop COLUMN 삭제할 컬럼명;
```

5. 테이블 삭제

```
drop table 테이블명;
```

6. 테이블의 모든 레코드행을 제거

```
truncate table 테이블명;
```

7. 테이블 명 변경

```
rename 기존테이블명 to 바꿀 테이블명
```

8. 현재 접속중인 사용자로 사용할 수 있는 테이블 명을 알고자 할 경우

```
select table_name from user_tables
order by table_name desc;
```

1. insert문

1. insert문: 테이블에 새로운 데이터를 저장하기 위해서 사용하는 데이터 조작 쿼리문이다.

형식)

```
insert into  테이블명
(컬럼명,컬럼명,....)
values(값,값,...);
```

가. 테이블명 다음에 기술한 컬럼 목록 순서대로 values 에 지정한 값들이 차례로 저장된다.

나. 테이블 명 다음에 컬럼명을 생략하면 테이블 생성시(create table) 지정한 컬럼명 순서대로 값들이 저장된다.

다. 테이블명 다음의 컬럼 목록 개수와 values 다음의 값의 개수가 일치해야 한다.

2. insert ALL문

가. **서브 쿼리의 결과를 조건없이 여러 테이블에 동시에 저장할 경우 사용한다.**

3. insert all when 조건식 then문

가. insert all when 조건식 then문:
복수개의 테이블에 다중행 레코드를 저장할 때 when 조건식에 지정한 조건에 맞는 자료만 저장 시킴.

1. update 문

1. update 문은 테이블에 저장된 자료를 **수정**하기 위해
사용하는 쿼리문이다.

2.update 문 문법구조
형식) **update 테이블명**
set 필드명=변경값,필드명=변경값, 중략
where 조건식;
조건식 맞는 레코드 만 수정

3.update문에서 **where 조건식을 주지 않으면 모든 행 레코드가 수정**된다.

2. **서브 쿼리를 사용한 데이터 수정**하기

1. update 수정문의 set절에 서브 쿼리문을 기술하여 수행한 결과로 자료를 변경한다. 이러한 방법으로 다른 테이블에 저장된 레코드를 이용하여 해당 컬럼 값을 수정할 수 있다.

2. 문법 형식)
update 테이블명
set (컬럼명,컬럼명,...중략) =(서브 쿼리)
where 조건식;

3. 테이블을 합병하는 MERGE

1. MERGE는 합병이라는 의미이며, 구조가 같은 2개의 테이블을 하나의 테이블로 합치는 기능을 한다.

2. MERGE 명령어를 수행할 때 수행하는 테이블에 기존에 존재하는 레코드가 있다면 새로운 값으로 UPDATE 되고, 존재하지 않으면 새로운 레코드로 INSERT 된다.

4. delete 문

1.delete문은 테이블에 저장된 **데이터를 삭제하는 쿼리문**이다.

2.delete문 문법 형식
형식)
delete from 테이블명
where 조건식;

3. delete문에서 **where 절을 사용하지 않을 경우 테이블에 있는 모든행이 삭제된다.그러므로 신중히 이 문을 사용**해야 한다.

주의사항) SELECT 검색시 오라클은 영문 레코드는 영문대소문자를 구분한다.

1. **트랜 잭션**

1. **트랜잭션이란?**
가. **트랜잭션은 데이터 처리의 한 단위를 의미한다.**
오라클에서 발생하는 여러 개의 sql문을 하나의 논리적 작업 단위로 처리하는 것을 말한다.

2. commit 이란?
가. **모든 sql문을 정상적으로 처리**하겠다는 뜻

3. ROLLBACK 이란?
가. **SQL문 작업을 취소** 하겠다는 뜻.

4. AUTO COMMIT(자동 커밋) 되는 쿼리문 정리
create(테이블 생성문), alter(테이블 수정문),
drop(테이블 삭제문),rename(테이블 명 변경문),
TRUNCATE(테이블 전체행 삭제문)

5. insert,delete,update을 데이터 조작어라 한다.
데이터 조작어는 auto commit이 아니다. 하지만
delete문을 실행하고 commit을 하지 않아도 auto
커밋 되는 create 문을 실행시 commit을 사용하지

않아도 delete문까지 오토커밋되어 버린다. 주의

6.데이터 조작어를 실행하고,auto commit되는 쿼리문
을 실행할때 에러가 발생해도 이전 데이터 조작어에
commit이 반영되어 rollback처리가 안된다.

7. SAVEPOINT 는 오라클에서 제공하는 기본 트랜잭션
범위를 인위적으로 작게 분할하는 것을 말한다.

8. 세이브 포인트에 의해서 지정한 세이브 포인트명
까지 이동할려면 rollback to 세이브 포인트명으로
처리 할 수 있다.그러면 이동한 세이브 포인트명까지
롤백 측 쿼리문 취소를 할 수 있다.

9. 세이브 포인트명 지정하는 형식
가. SAVEPOINT 세이브포이트명;

10.세이브 포인트명으로 지정된 곳 까지 이동 즉
되돌아 가는 형식
형식)
ROLLBACK TO 세이브 포인트명;

1. 트랜 잭션과 잠금

1. 오라클은 여러명의 사용자가 동시에 하나의 테이블에
접근하여 DML(insert,update,delete) 조작시 특정 사용자
가 자원을 독점하지 못하게 하기 위해서 락(잠금)을 발
생시킨다.

2 . 이러한 락을 해제 할려면 트랜잭션을 처리해야한다.
곧 commit,rollback 처리를 해야 한다.

3. 데드락(Dead Lock) 이란?
가. 서로 락 즉 잠금 상태가 되어져서 무한 대기 상태
인 경우를 Dead Lock이라 한다. 오라클에서 dead
lock이 걸리면 자원을 사용할 수 없는 상태가 된다.

나. 이러한 데드락이 발생하면 오라클은 기본으로

오라클을 사용하지 못하는 상황을 방지하기 위해서 다음과 같은 메시지를 보내면서 비 정상적인 종료를 하게 한다.

메시지) ORA-00060:자원 대기중 교착 상태가 검출되었습니다.

1. TRUNCATE와 delete문의 차이점

1.TRUNCATE 문의 특징

가. TRUNCATE문은 AUTO COMMIT 문이다. 즉 트랜잭션에 의한 커밋을 처리하지 않아도 삭제 쿼리문이 반영된다.

나.삭제후 롤백에 의한 자료 복구가 불가능하다.

다. 전체 행 레코드는 삭제할 수 있지만 WHERE 조건절에 의한 조건에 맞는 레코드만 삭제하는 것은 불가능하다.

2.delete문의 특징

가. delete문 실행후 반드시 커밋 또는 롤백에 의한 트랜잭션 처리를 해야한다.

나. 롤백에 의한 데이터 복구가 가능하다.

다. 전체 행 레코드 삭제도 가능하고, where조건절에 의한 조건에 맞는 레코드만 삭제하는 것도 가능하다.

제약 조건

1. **데이터 무결성 제약 조건이란 테이블 컬럼에 부적절한 자료가 저장되는 것을 방지하기 위해서 테이블 생성시 각 컬럼에 대해서 정의하는 여러가지 규칙**을 말한다.

2. 제약조건 종류

가.**기본키(primary key): 중복 자료를 금지하고, null이 저장되는 것을 방지**한다. 유일한 값을 가지고, 반드시 자료를 입력해야 한다는 제약조건이다.

나. **not null 제약조건**: 컬럼에 **반드시 값을 저장해야 한다.** 즉 **null이 저장되는 것을 방지**한다.

제약조건명, 제약 조건의 유형, 제약조건이 속한 테이블명을 확인하는 쿼리문

```
select constraint_name,constraint_type,
table_name from user_constraints;
```

1. constraint_name 컬럼에는 **제약조건명**

2. constraint_type에는 **제약 조건 유형**

3. **제약조건 유형** 의미

P	기본키(Primary key)
C	not null+check제약조건
U	unique(중복값이 없고,null을 허용하는 제약조건)
R	외래키(Foreign key)

4. table_name은 **제약조건이 속한 테이블명**

unique 제약조건 특징

1. **특정 컬럼에 자료가 중복 되지 않게 한다.**

2. null 저장되는 것은 허용한다.

3. 중복 자료에서 null은 체크하지 않는다. 즉
null은 중복을 허용하는 특징이 있다.

CONSTRAINT 키워드로 정의하는 사용자 정의
제약조건명 설정법

1. 오라클은 컬럼 설계시 제약조건 유형을 설정
하면 기본으로 제공하는 제약조건명을 자동으로
부여한다.
기본 제약조건명 형식) SYS_숫자번호형식

2.사용자 직접 constraint 키워드로 사용자 정의
제약조건명을 설정할 수 있다.

3.사용자 정의 제약조건명 설정 형식
형식)
컬럼명 자료형(크기) constraint
사용자정의 제약조건명 제약조건유형(not null)

4. 사용자 정의 제약조건명 명명 규칙
형식)
테이블명_컬럼명_제약조건유형

CHECK 제약조건

1. check 제약 조건은 조건에 맞는 자료만 저장
되게 한다.

제약조건 지정방법

1.컬럼 레벨 지정법 : 컬럼에 직접 제약조건을 지정
한다.

2. 테이블 레벨 지정법

가. 컬럼을 모두 정의하고 나서 테이블 정의를 마무리 짓기 전에 일괄적으로 한꺼번에 제약조건을 지정하는 방법이다.

나. 하나의 테이블에 2개 이상의 기본키를 설정하는 것을 복합키라 한다. 이러한 복합키는 반드시 테이블 레벨 지정법으로 정의 해야 한다.

다. not null은 컬럼 레벨 지정법으로 직접 컬럼 생성시 제약조건을 정의해야 한다.

not null 제약조건 추가

1. not null 제약조건을 추가하기 위해서는 add 대신 modify문을 사용한다. 이유는 null을 허용한 상태에서 not null로 수정한다는 뜻이 있기 때문에 modify문을 사용한다.

형식)

```
alter table 테이블명
modify 컬럼명
constraint 사용자정의 제약조건명 not null
```

제약조건 제거 형식

```
alter table 테이블명
drop constraint 제약조건명;
```

제약조건 비활성화 특징

1. 제약 조건을 삭제하지 않고 비활성화 하여 잠시 사용을 보류하는 것을 말한다.

2. 제약조건 비활성화 형식

alter table 테이블명
disable constraint 제약조건명;

제약조건 활성화 형식

1. 잠시 비활성화 되어져서 사용이 보류된 제약조건을 다시 사용할려고 활성화 하는 것을 말한다.

2. 제약조건 활성화 형식

alter table 테이블명
enable constraint 제약조건명;

CASCADE옵션 특징

1. CASCADE 옵션은 부모 테이블과 자식 테이블 간에 기본키와 외래키 참조 관계가 설정된 경우 부모 테이블의 기본키 제약조건을 비활성화 시키면 연속적으로 자식 테이블의 외래키 제약조건까지 함께 한번에 비활성화 시키는 법.

가상테이블(뷰:VIEW) 이란?

1. 가상테이블 뷰는 실제 테이블에 근거한 논리적인 가상 테이블이라 정의할 수 있다.

2. 가상테이블 뷰에는 실제 자료가 저장되지 않는다. 마치 테이블 처럼 동일하게 뷰를 통해 실제 테이블의 레코드를 볼 수 있다.

3. 가상테이블 뷰를 통해 사용자에게 실제 테이블 사용을 제한할 수 있다.

4. **뷰 생성 문법 형식**
   create view 뷰이름
   as
   서브 쿼리문;

```
select view_name,text
from user_views;
```

1. **view_name 컬럼**에는 생성된 **가상테이블 뷰이름을 확인**할 수 있다.
   2. **text 컬럼**에서는 create view에 의한 뷰생성시 as문 다음의 **서브 쿼리문이 저장**된다. 그러므로 **뷰에는 실제 자료가 저장되지 않고,뷰 생성시 작성한 서브쿼리문이 저장된다는 것을 알 수 있다.**

가상테이블 뷰를 사용하는 이유

1. **뷰를 사용하는 이유는 복잡하고 긴 쿼리문을 뷰로 정의하면 접근을 단순화** 할 수 있기 때문이다.

오라클 사용자 생성법

1. 오라클 DB 사용자를 만들려면 dba 즉 system **관리자로 접속**

2. SQL> **CREATE USER 사용자명**
   identified by 비밀번호;
   사용자와 비번을 생성한다.

3. **사용자만 생성해서는 오라클에 접속할 수 없다.**

4. **오라클 접속권한**
   가. **grant create session to 사용자명**

:오라클에 접속할수 있는 권한을 설정

5. 테이블 또는 가상 뷰테이블 검색권한 설정
   가. grant select on 테이블명(가상뷰테이블명)
       to 사용자명;

가상테이블 뷰 삭제 문법

1. 형식) drop view 뷰이름;

가상테이블 뷰 생성 옵션 문법

1. create or replace view 뷰이름;
   or replace옵션은 존재하지 않는 뷰생성시
   새로운 뷰를 만들게 하고,기존에 존재하는 뷰인
   경우는 내용을 수정하게 한다.

2. create or replace FORCE view 뷰이름;
   가. FORCE옵션은 기존 테이블이 존재하지 않아도
      뷰를 생성하게 한다.

   나. 이 옵션을 생략하면 기본값이 NOFORCE가된다.
       NOFORCE 옵션은 반드시 기존 테이블이 있어야
       가상테이블 뷰를 생성할 수 있다.

가상 뷰 생성시 사용하는 with check option

1. 뷰를 생성할 때 조건 제시에 사용된 컬럼값을
  변경 못하게 한다.

with read only 옵션

1. 이 옵션을 사용해서 **뷰를 만들면 가상테이블**
뷰를 통한 기존 테이블의 어떤 컬럼 레코드값도
수정 못하게 한다.

오라클에서 내부적으로 제공하는 ROWNUM 컬럼의 특징

1.**ROWNUM 컬럼은 오라클에서 내부적으로 제공하는**
컬럼명이다.

2. **ROWNUM 컬럼은 테이블에 레코드 저장시 저장된**
순서에 따라 1부터 1씩증가되는 일련의 번호값이
차례대로 저장된다. 이 **컬럼값은 자료가 입력되는**
시점에 결정되기 때문에 정렬 검색 할때
바뀌지 않는다.

3. **ORDER BY 정렬에 의해서 다시 정렬된 순서대로**
ROWNUM컬럼값을 변경하고자 할 경우는 새로운 테이
블에서 다시 작업을 해야한다. 이런 경우는 테이블
의 저장 공간이 필요하기 때문에 **가상 뷰테이블을**
사용하면 효과적이다.이유는 가**상 뷰테이블은**
저장 공간이 필요 없기 때문이다.

인라인 뷰란?

1. **인라인 뷰는 sql문에서 사용하는 서브 쿼리문**
의 일종으로 **from 절다음에 위치하여 마치**
테이블 처럼 사용된다.

2. **인라인 뷰는 메인 쿼리문의 select 문 from**
절 다음에 사용되는 서브 쿼리문을 말한다.

3.사용문법 형식

```
select ...
from ..(select .. )
```

where 조건식;

오라클 인덱스 특징

1.오라클에서 인덱스를 사용하는 이유는 **보다 더 빠른 검색을 위해서 사용**한다.

2. **기본키 또는 유일키 제약조건으로 지정된 곳은 오라클에서 해당 컬럼에 자동으로 인덱스를 생성**해 준다.

3. set timing on : **인덱스에서 검색 속도를 체크**할 수 있다.

4.**인덱스 생성 문법**

형식)

```
create index 인덱스명
on 테이블명(컬럼명);
```

5.**인덱스 삭제** 문법

형식)

```
drop index 인덱스명;
```

고유 인덱스 특징

1. **고유 인덱스는 기본키 또는 유일키 처럼 유일한 값을 갖는 컬럼에 대해서 생성하는 인덱스**를 뜻한다.

2. **중복자료가 있는 컬럼에는 고유 인덱스를 설정 못함.**

3.**고유 인덱스 생성 문법**

형식)

```
create unique index 인덱스명
on 테이블명(컬럼명);
```

비고유 인덱스 특징

1. **중복된 자료가 저장된 컬럼에 대해서 생성하는 인덱스를 비고유 인덱스**라 한다.

2. **중복자료가 있는 컬럼에는 비고유 인덱스만 설정 가능**하다.

시퀀스

1. **시퀀스는 중복번호가 없고,null이 발생되지 않는다.**

2.**시퀀스를 사용할 컬럼은 반드시 기본키 제약조건과 오라클 정수형 자료형 타입으로 선언**해야 한다.

시퀀스 생성 문법

1.시퀀스 생성: **create sequence 시퀀스명**;

2.시퀀스 생성시 **부가적인 옵션** 명령어:

가. **increment by 1** : 시퀀스 생성시 1씩 증가되는 시퀀스를 생성

나.**start with 1** : 1부터 시작되는 시퀀스를 생성하는 부가 옵션

다.**nocache** : 이 옵션을 사용하면 시퀀스를 임시 메모리 상에서 사용하지 않겠다는뜻. 기본값은 20, cache는 메모리에 시퀀스값을 미리 할당해 놓음. minvalue는 최소값, maxvalue는 최대값, nocycle은 기본값으로 최대나 최소값에 도달하면 생성 중지, cycle은 증가는 최대값에 도달하면 다시 최소값부터 시작, 감소는 최소값에 도달하면 다시 최대값부터 시작.

3. 현재 만들어진 시퀀스 이름,증가값을 확인

```
select sequence_name,increment_by
from user_sequences;
```

가.sequence_name 컬럼값은 시퀀스 이름을 저장
나.increment_by 컬럼값은 각 시퀀스의 증가값에
대한 정보를 저장하고 있다.

4. 가 .CURRVAL : 시퀀스로 부터 현재값을 가져온다.
나. nextval : 시퀀스로 부터 다음값을 가져온다.

5. 새로 만든 시퀀스로 nextval을 하지않고 currval
로 현재값을 가져오면 오류가 발생한다. 이유는
nextval로 새로운 값을 생성하지 않았기 때문이다.

6.CURRVAL 특징

가. currval 은 nextval 후에 같은 세션에서만 현재 시퀀스 번호를
가져오는 것이 가능하다.

나. nextval 을 사용하고 난 다음에 그 세션이 종료하기 전까지
currval 로 값을 가져올수 있다.

다. 시퀀스명.currval값은 한 session에서만 존재하는 임시값이다.
currval값은 세션에서 마지막으로 call한 nextval에 의해 리턴된 값에
의해 정의 된다.
만일 세션에서 nextval값이 아직 call되지 않았다면
currval 값은 정의 되지 않는다.
한 세션에서 nextval을 먼저해야 currval을 할수 있다.

오라클 사용자 생성 문법

1.형식:

```
create user 사용자명
identified by 비밀번호;
```

사용자 권한 설정 형식

1.형식:
grant 권한 to 사용자명;

2. grant 권한 to 사용자명
with admin option;
=>사용자 권한 설정시 부가옵션으로 with admin option 권한을 설정하면 권한이 설정이 된 사용자가 데이터베이스 관리자(DBA)가 아니어도 자신이 부여 받은 권한을 다른 사용자에게 부여할 수 있는 권한도 함께 부여된다.

3. 오라클 DBA(데이터베이스 관리자)가 설정하는 시스템권한 종류
가. 사용자 생성후 grant create session to 사용자; 로 create session 권한을 설정해야 생성된 사용자가 오라클에 접속할 수 있다.

나.grant create table to 사용자; 생성된 사용자에 create table 즉 테이블 생성권한을 할당해야 한다. 하지만 테이블 생성권한만 주어서는 테이블 생성을 할 수 없다.그러므로 각 사용자에게 테이블 사용 공간인 테이블 스페이스를 할당해야 한다.

각 객체 즉 테이블을 소유한 사용자가 객체의 모든 권한을 가지고 있다.
1.객체 권한 설정형식
grant 권한 on 테이블명 to 사용자명;

```
select grantee,table_name,grantor,privilege
from user_tab_privs_made;
```

1. grantee 컬럼: 권한이 할당된 사용자 명

2. table_name:각 객체,가상테이블 뷰 이름
3. grantor 컬럼:권한을 준 사용자명
4. privilege 컬럼:권한 이름 테이블 select 권한..

사용자에게 부여한 각 객체 권한을 철회하기 위한 명령어 형식

형식) REVOKE 권한 ON 테이블명 from 사용자명;

grant 권한 on 테이블명 to 사용자
with grant option;

=>사용자에게 객체 권한을 with grant option과 같이
부여하면 부여받은 사용자는 그 권한을 다른 사용자
에게 부여할 수 있는 권한도 함께 주어진다.

롤(ROLE) 이란?

1.롤은 사용자에게 보다 더 간편하게 권한을 부여할
수 있도록 여러 개의 권한을 묶어 놓은 권한의 집합
을 뜻한다.

롤(ROLE)의 생성 절차

1. 롤을 생성하기 위해서 DBA(system)계정으로 로그

인

2.롤을 생성
형식)create ROLE 롤이름;

3.생성될 롤에 권한을 부여
형식)부여되는 권한이 시스템권한(create session
(오라클 연결 권한),create table,create view)
일때는 DBA로 접속에서 부여

grant create session,create table,create
view to 롤이름;

형식)부여되는 권한이 객체 권한일때는 객체 소유
자로 접속해서 부여.
grant 객체권한 to 롤이름;

4. 사용자에게 생성될 롤을 부여하는 작업(DBA로
접속)
형식)grant 롤이름 to 사용자;

5. 사용자에게 생성된 롤을 확인 방법
형식)
```
select username,granted_role
from user_role_privs;
```
가.username 컬럼에는 사용자명
나.granted_role 컬럼에는 사용자에게 설정된
롤이름이 저장.

롤회수 란?

1. 롤(ROLE)회수는 특정 사용자에게 해당 롤을 사용
할 수 없도록 회수하는 것을 말한다. 해당롤은 존재하
기 때문에 다른 사용자게 롤을 부여할 수 있다.

2.롤 회수 형식문법
형식) REVOKE 롤이름 FROM 오라클 사용자명;

롤 제거란?

1. 롤 제거는 롤 자체를 삭제하는 것을 말한다.

2. 롤자체가 제거 때문에 다른 사용자에게 롤을 부여
   할 수 없다.

3. 롤 제거 형식 문법
   형식) DROP ROLE 롤이름;

오라클 저장프로시저

1. 복잡한 쿼리문을 매번 사용할때 다시 입력할 필요 없이 간단하게 저장프로시저로 정의해 놓고
 호출해서 복잡한 쿼리문에 대한 실행 결과를 얻을려고 할때 주로 사용한다.

 2. 저장프로시저를 사용하면 성능도 향상되고, 호환성 문제도 해결된다.

 3. 저장프로시저 사용 문법

```
create or replace procedure sel_board13
--> or replace 는 같은 이름의 저장프로시저를 생성할 경우 기존 프로시저를 삭제
-->하고 새롭게 기술한 내용으로 재생성 하는 옵션
-->sel_board13 은 저장프로시저 이름
( vname out board13.name%TYPE,
  vtitle out board13.title%TYPE,
  vcont out board13.cont%type,
  vnum in board13.num%type
)
-->mode 매개변수라고 한다. mode 매개변수의 종류는 3가지가 있다.
```

-->in은 값을 전달받을때 사용,out는 디비 레코드값을 되돌려 받을때 사용한다. 즉

-->출력 결과물을 받을때 사용한다. inout는 두가지 목적에 모두 사용할 경우 사용한다.

```
is
begin
  select name,title,cont into vname,vtitle,vcont
  from board13
  where num=vnum;
end;
/
```

-->begin 과 end 사이에 실제 실행할 쿼리문 문장이 들어가면 된다.

4. 저장프로시저 만드는 순서

가. sqlplus로 접속

나. sql프롬프트>ed 저장프로시저로 작성할 sql 스크립트 파일명 fun02 로 입력

다. 메모장이 열리면 저장프로시저 생성 쿼리문을 입력한다.

라. @fun02 를 입력해서 저장 프로시저를 생성

마. sql>execute 저장프로시저이름(전달될값); 으로 실행한다. 이전 실행명령어를 취소하고 싶으면 트랜잭션의 rollback; 하면 쿼리문이 실행이 취소된다. 반대로 commit;

하면 저장프로시저 실행이 성공적으로 완료된다.