

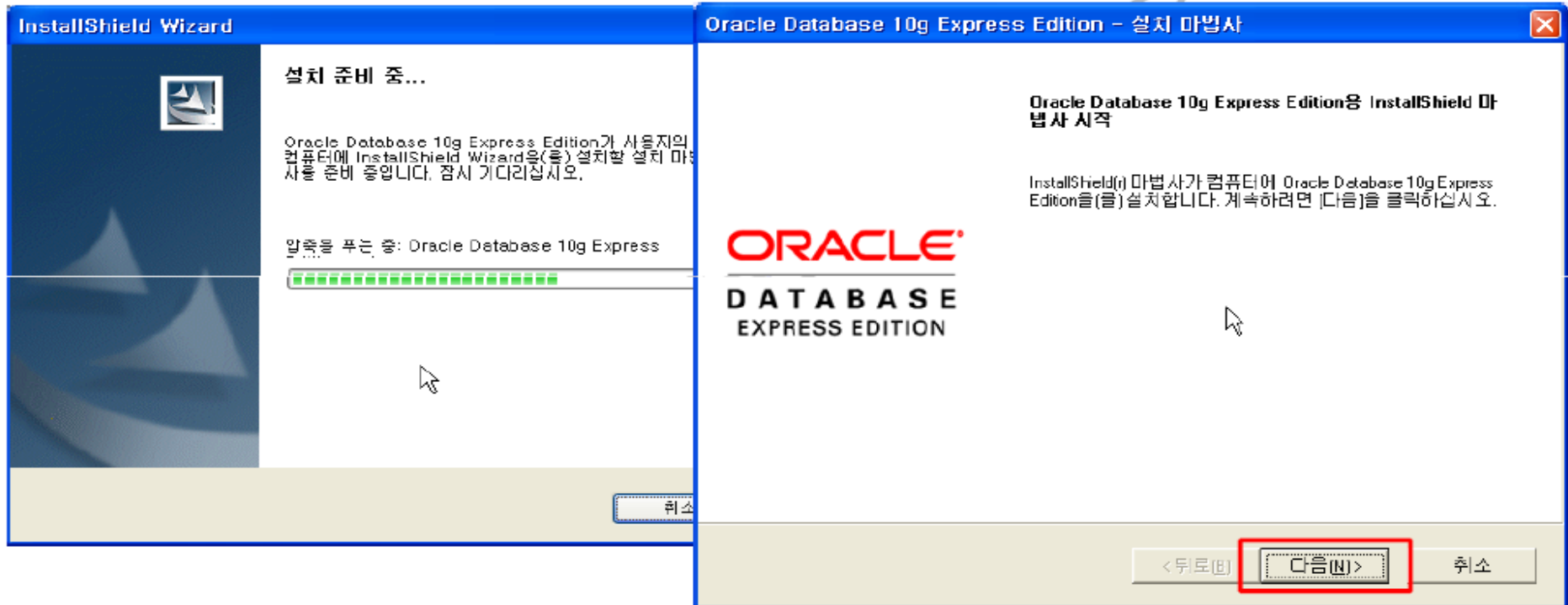
# Oracle(SQL)

Oracle 설치



# Oracle 설치

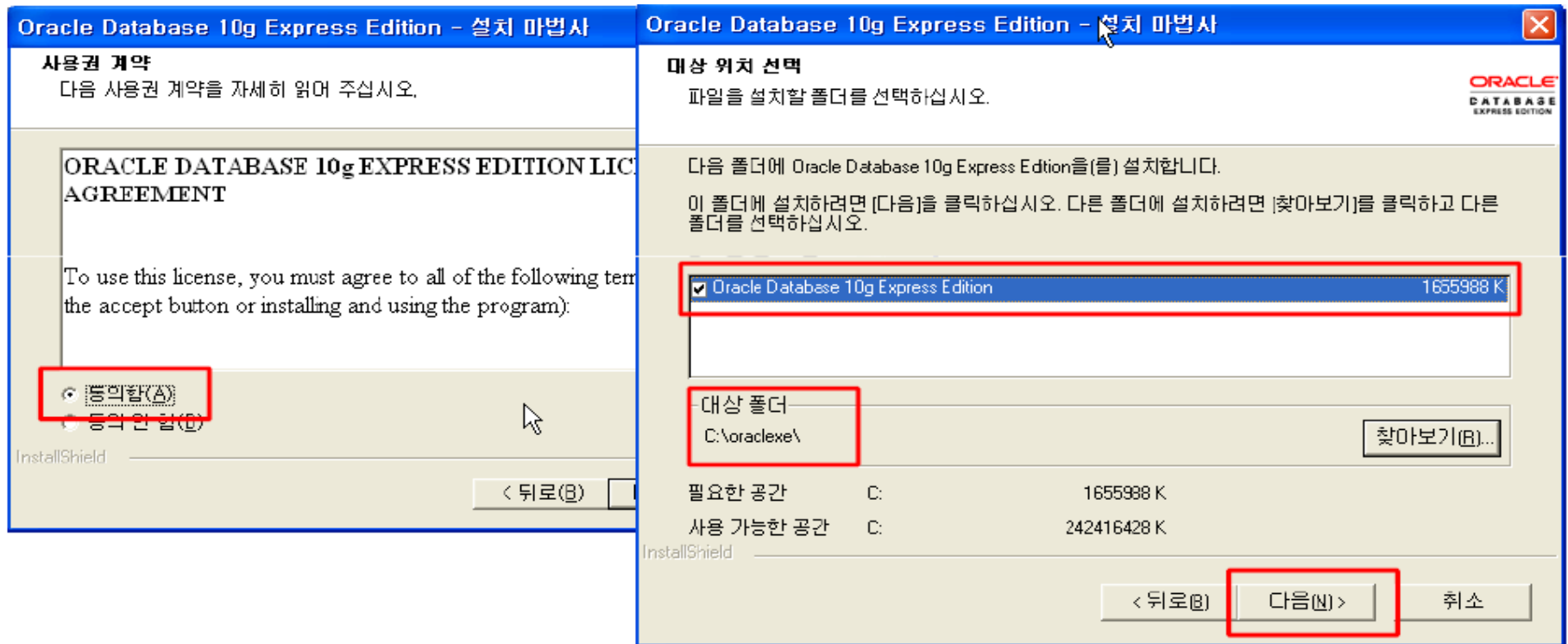
- OracleXEUniv.exe 파일을 더블 클릭하여 설치를 시작한다.



# Oracle 설치

카페

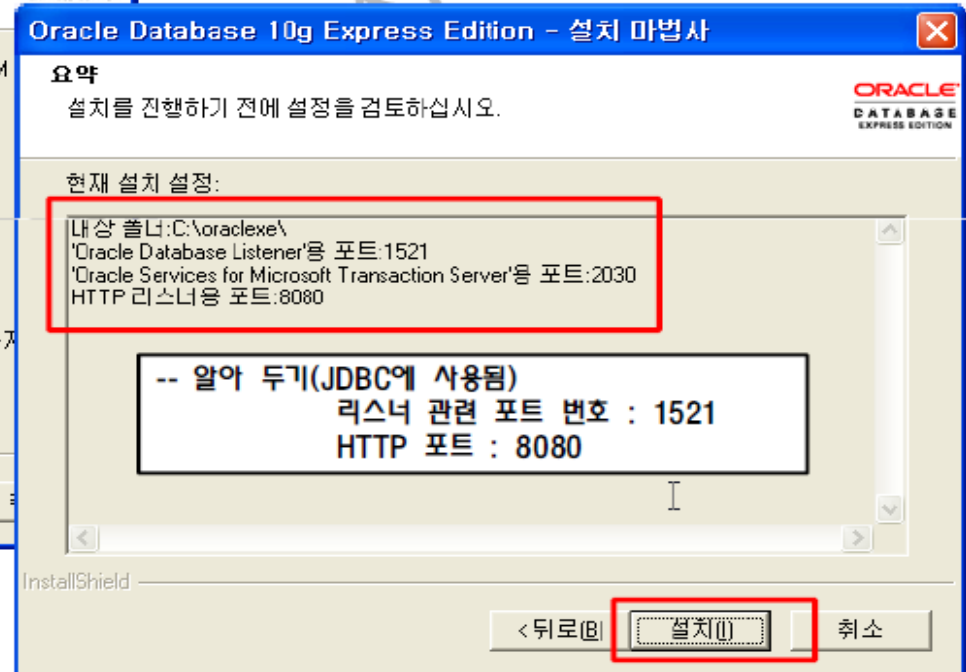
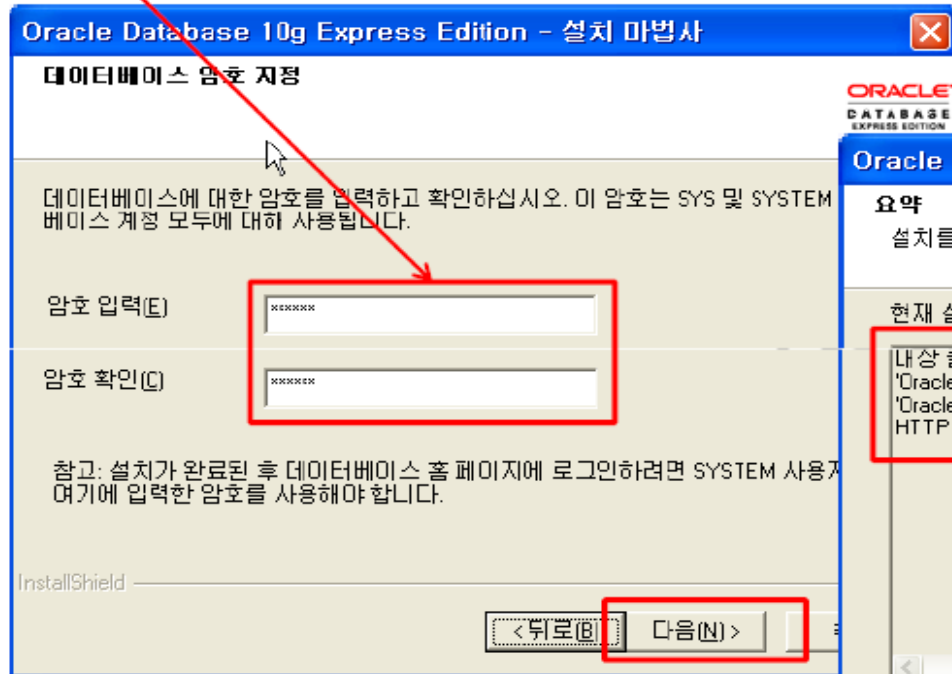
- [동의함] 버튼을 클릭한 후 해당 설치 폴더를 확인하도록 한다.
- 기본 경로는 [c:\oraclexe] 폴더이다.



# Oracle 설치

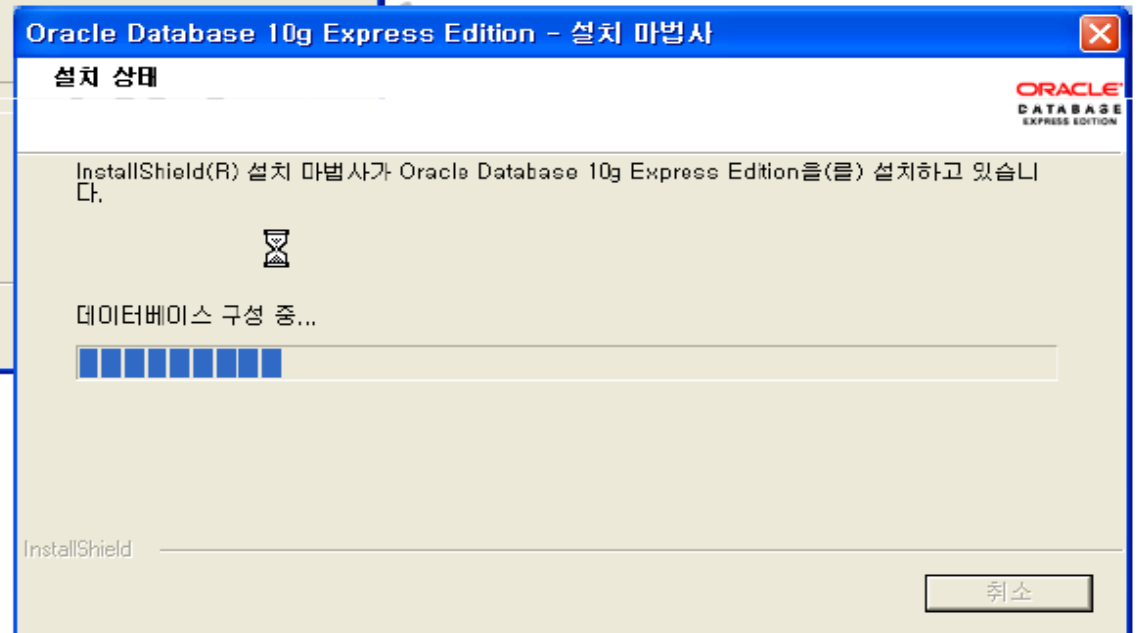
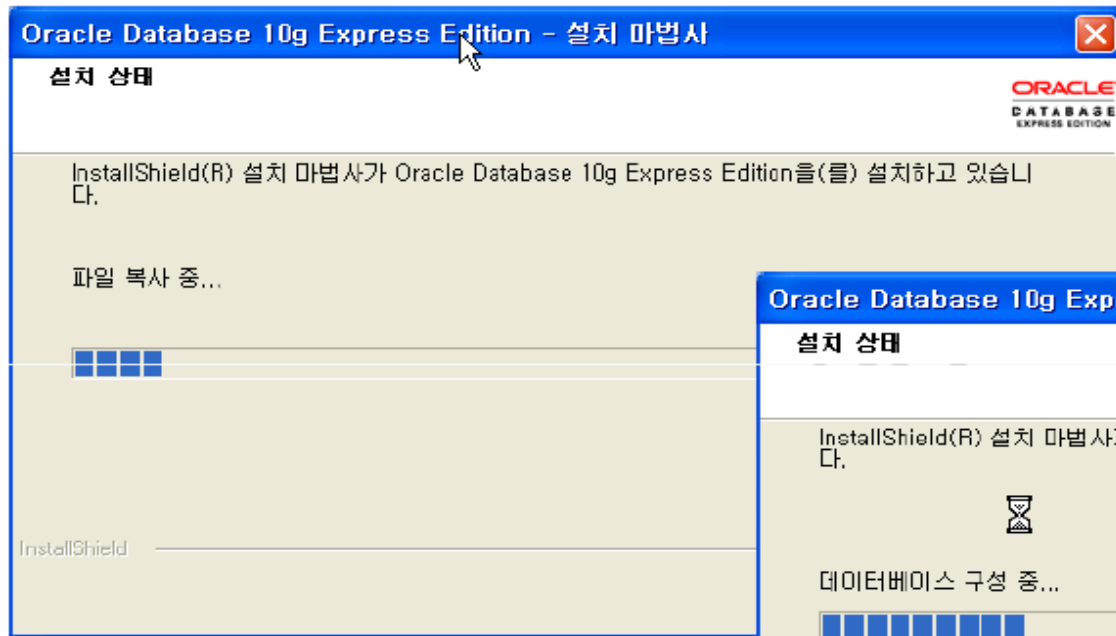
카페

- 암호 : oracle 으로 입력하도록 한다.(다른 것으로 설정해도 상관 없다.)



# Oracle 설치

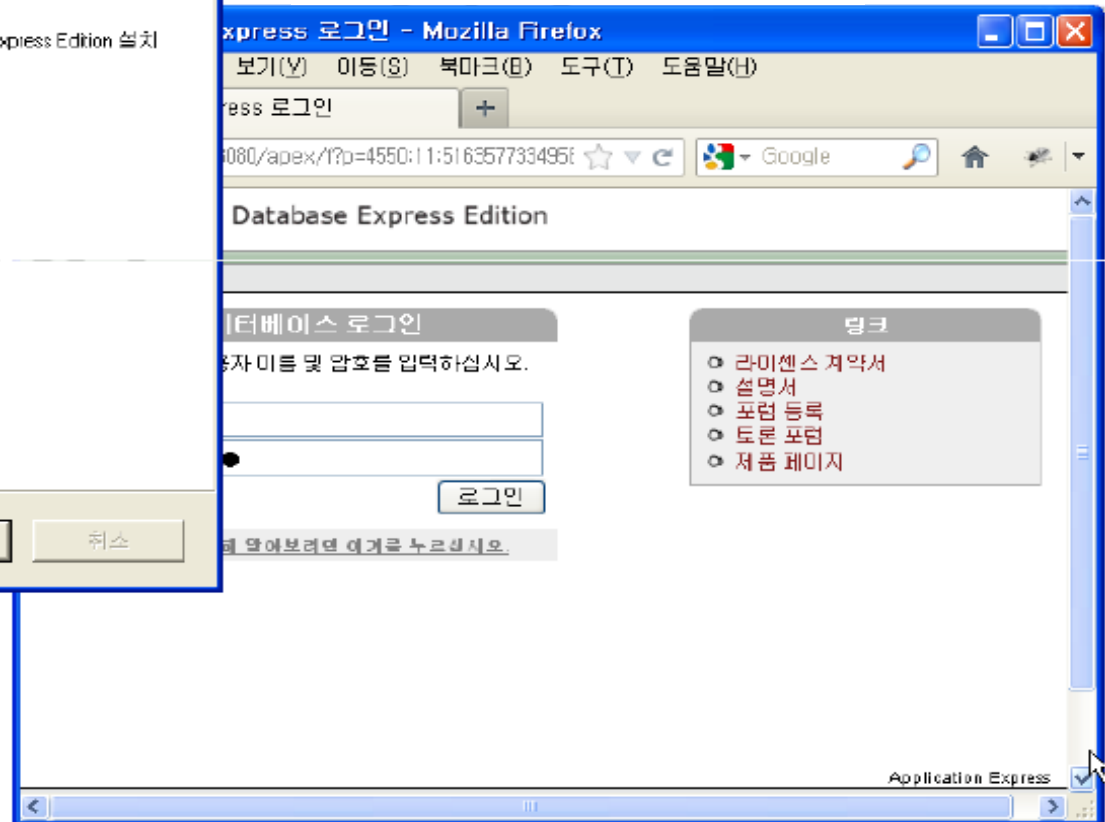
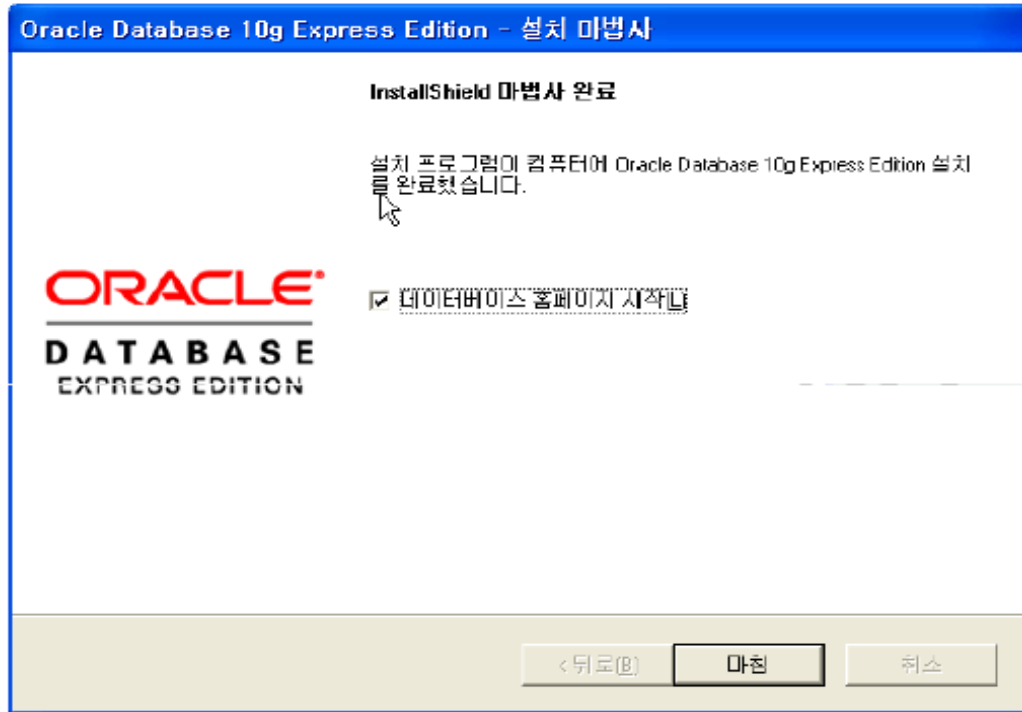
- 다음과 같이 파일이 복사된다.



# Oracle 설치

카페

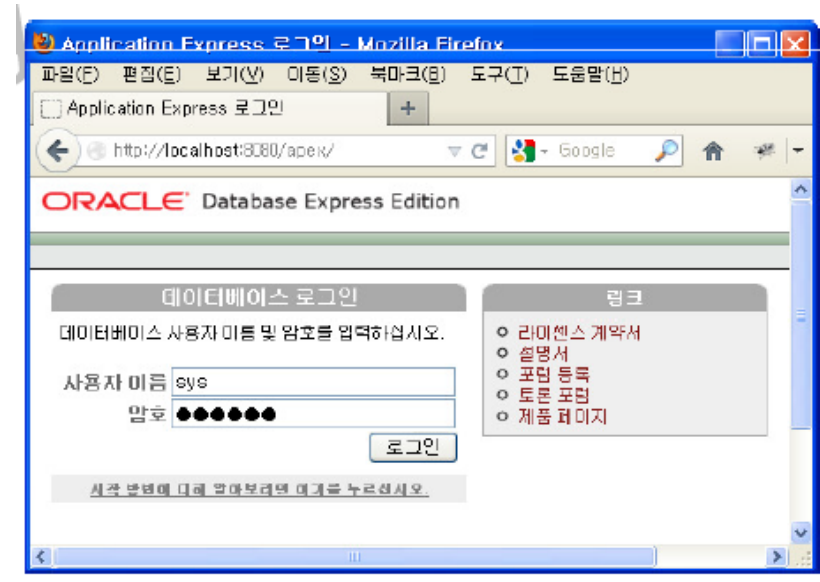
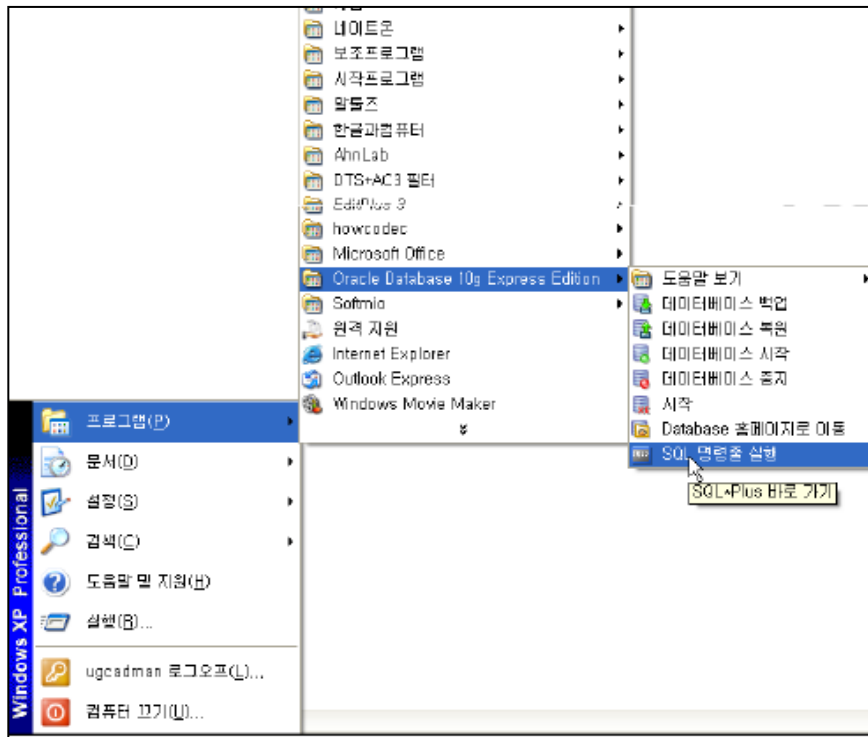
- 정상적으로 설치가 되었고, 시작 홈페이지가 로딩된다.



# Oracle 설치

보류

- Oracle의 기본 포트는 8080이다.
- 접속 방법
  - 웹브라우저: `http://localhost:8080/apex` 으로 접속하면 된다.
  - 명령줄 실행 : sql 명령줄 실행



# Oracle(SQL)

데이터베이스 개요





# 데이터베이스와 DBMS

카페

- 데이터 베이스 객체(Object)

객체 이름	설명
테이블	행과 열로 구성된 2차원적인 표
시퀀스	정수형 번호표 생성기
인덱스	데이터 검색을 빨리 하기 위하여 만들어 둔 개념
프로시저	반환 타입이 없는 객체
함수	반환 타입이 있는 객체

- 데이터 베이스 객체 현황 보기

- conn / as sysdba (관리자 모드로 로그인 하는 명령어)
- select distinct object\_type from dba\_objects
- order by object\_type ;

# 데이터베이스와 DBMS

카페

- [Database]는 연관된 데이터를 유기적으로 묶어둔 객체를 의미한다.
- 객체
  - 데이터 베이스를 구성하고 있는 요소들을 의미한다.
  - 테이블, 시퀀스, 인덱스, 시너님, 뷰 등등을 말한다.

SQL 타입	타입
데이터베이스의 목표	지속적인 데이터의 관리 및 보호 안전성 보장 무결성 보장
데이터베이스의 특징	실시간 접근이 가능해야 한다. 동적인 변화 동시에 공유 가능 내용에 의한 참조 가능
DBMS	Database Management System 데이터베이스를 관리하는 시스템 또는 프로그램 mySql, MS-SQL, Oracle, DB2 등등

# 데이터베이스와 DBMS

카페

- DBMS의 주요 기능
  - 데이터의 추가/조회/변경/삭제 기능이 구현되어야 한다.
  - 데이터의 무결성(신뢰도) 유지되어야 한다.
  - 트랜잭션 관리 기능이 구현되어야 한다.
  - 데이터의 백업 및 복원이 가능하다.
  - 데이터 보안 기능이 되어야 한다.
- DBMS 필수 기능은 다음과 같다.

내용	설명
정의 기능	데이터 베이스의 논리적, 물리적 구조를 정의하는 기능
조작 기능	검색, 갱신, 삽입, 삭제
제어 기능	정확성과 안전성을 유지하는 제어 기능

# SQL이란?

카페

- Ddtabase로부터 데이터를 조회/삭제/수정 등의 작업을 수행할 때 사용하는 질의 언어를 말한다.
- Strutured Query Language(구조적 질의 언어)
  - 질의(Query) : 질문을 던진다.
  - 데이터베이스에게 질문을 하여 특정 결과를 얻어 내거나 특정 작업을 수행하는 것
  - 예시
    - 100번 사원의 급여는 얼마인가요?
    - 데이터 3건을 수정하세요.
- SQL = DDL + DML + DCL + DQL + TCL + ...

구문	이름	키워드 및 설명
DDL	데이터 정의어	CREATE(생성), ALTER(변경), DROP(제거)
DML	데이터 조작어	데이터 삽입(insert)/수정(update)/삭제(delete)
DCL	데이터 제어문	GRANT, REVOKE(사용자 관리/권한 주기 등등)
DQL	데이터 질의어	데이터 조회(select 문장)
TCL	트랜잭션	COMMIT, ROLLBACK

# SQL이란?

카페

구문	문법 요소	설명	주체	트랜잭션
DQL	select	데이터 질의어	-	필요 없음
DML	insert update delete merge	Data manipulation language 데이터 조작어	row 단위	commit ; rollback ;
DDL	create alter drop rename truncate comment	Data definition language 데이터 정의어	Object 단위	Auto Commit
DCL	grant revoke	Data control language 데이터 제어어	user(사용자)	Auto Commit
TCL	commit rollback savepoint	Transaction control	-	-

# SQL의 데이터 타입

카페

Type	Description
NUMBER, NUMBER(숫자), NUMBER(숫자1, 숫자2)	숫자형 데이터 <정수 / 실수 가능>
CHAR	문자열 저장 타입. 최대 2000 바이트까지 가능.
VARCHAR2	가변길이 문자열 타입. 최대 4000 바이트 까지 가능.
DATE	날짜 데이터 타입. 시스템 설정에 따라 출력 형태가 다르다.
NCHAR	유니코드타입의 CHAR. 최대 2000 바이트. 디폴트는 1바이트다.
NVARCHAR2	유니코드 가변 CHAR. 최대 4000 바이트까지 가능하며 반드시 길이를 정해야한다.
LONG	가변길이 CHAR 타입으로 최대 2GB 까지 가능하다.
RAW	가변길이 바이너리 데이터로 최대 2000 바이트까지 가능하다.
LONG RAW	가변길이 바이너리 데이터로 최대 2GB 까지 가능하다.
BLOB	바이너리 데이터로 4GB 까지 가능하다.
CLOB	싱글-바이트 캐릭터 데이터로 최대 4GB 까지 가능하다.
NCLOB	모든 캐릭터 데이터로 최대 4GB 까지 가능하다.
ROWID	ROW의 물리적 주소를 나타내는 바이너리 데이터다.
TIMESTAMP	DATE 값을 미세한 초단위까지 저장한다. NLS_TIMESTAMP_FORMAT 으로 처리한다.
INTERVAL YEAR TO MONTH	두 DATETIME 값의 차이에서 년도와 월만 저장한다.
INTERVAL DAY TO SECOND	두 DATETIME 값의 차이에서 일, 시간, 분, 초 까지 저장한다.

# Table(테이블)

카페

- 테이블 : 행과 열로 구성된 2차원 형태의 자료 구조를 말한다.
- 행 = 로우 = Row = 레코드
- 열 = 컬럼 = Column(id, name, salary, birth, email)
- 스키마 - 테이블의 구조에 대한 정보 저장(메타 데이터)
- null 값이란 비교 판단이 불가능한 데이터를 의미한다.

ID	NAME	SALARY	BIRTH	EMAIL
-----				
1	제시카	100	2002-05-05	aaa@naver.com
2	티파니	200	2001-01-01	ddd@naver.com
3	수영	200	2001-05-05	bbb@naver.com
4	효연	400	2004-04-04	ccc@naver.com
5	서현	500	2004-08-08	fff@naver.com
6	유리	300	2004-04-04	ttt@naver.com
7	태연	800	2008-08-08	
8	윤아	0	2012-04-09	
9	써니	0	2012-04-09	

# 프라이머리 키와 인덱스

카페

- 프라이머리 키(Primary Key)

- 각각의 레코드(행)를 구별하기 위한 칼럼/필드들을 말한다.
- 각 레코드(행)의 고유 식별자를 의미한다.
- 프라이머리 키는 반드시 입력(not null)되어야 하고, 중복되어서는 안 된다.(unique)
- 프라이머리 키를 생성하게 되면, 내부에서 인덱스가 생성된다.
- 예시) 사원 테이블의 사번, 병원의 환자 코드

사번	이름	생일	부서
1	홍길동	1970/01/01	총무부
2	홍길동	1980/12/25	영업부
3	최민수	1970/01/01	구매부

- 인덱스(Index)

- 검색 속도의 향상을 위하여 특정 필드를 사용하여 레코드를 순서대로 정렬한 정보를 저장하고 있다.
- 예시) 교과서의 인덱스, 책의 찾아 보기 기능



# Oracle(SQL)

기본 설정하기



# Oracle 설치하기

보류

버전	설명
XE 버전	OracleXEUniv.zip 파일을 압축 해제 후 설치 scott.sql 스크립트 파일을 이용하여 사용자 생성.
Standard 버전	10g_oracle_for_winXP.zip 파일 압축 해제 setup.exe 파일을 더블 클릭하여 실행시킨다. 참고 사항 oracle 홈 위치 : c:\oracle\product\10.2.0\db_1 SID(DB 이름) 셋팅 전역 데이터베이스 이름 orclXXX (기본 값은 orcl) XXX : 자신의 아이피 번호 끝 3자리 숫자 DB 암호 설정 oracle, oracle으로 입력

# scott 사용자 만들기

카페

- Oracle xe 버전은 다음 내용을 수행하여 scott 사용자를 만들도록 한다.
- 다음 폴더에 가서 scott.sql 파일을 찾는다.
  - C:\oraclexe\app\oracle\product\10.2.0\server\RDBMS\ADMIN\
- Sql plus를 사용하는 경우
- c:\labs 폴더에 복사한다.(만들지 않았으면 생성 요망)
- sql 접속 창에서 다음과 같이 수행한다.
  - conn / as sysdba
  - @c:\labs\scott.sql
  - conn scott/tiger
  - 연결되었습니다. <-- 이 메시지가 뜨면 오케이

# 10g의 향상된 기능

보류

- 관리자로 접속하여 현재 접속한 사용자명으로 프롬프트 변경됨
  - SQL> set sqlprompt "\_user> "
  - SYS> conn scott/tiger
  - SCOTT>
- 사용자와 함께 권한을 표시
  - SQL> set sqlprompt "\_user \_privilege> "
  - SYS AS SYSDBA>
- 사용자와 함께 권한, 그리고 현재 시간을 같이 표시
  - SQL> set sqlprompt "\_user´ :´ \_privilege ´on´ \_date> "
  - SYS :AS SYSDBA on 12/19 14:11>
- 사용자명, 권한, 현재 날짜, 접속문자열 표시
  - SQL> set sqlprompt "\_user´ :´ \_privilege ´on´ \_date ´using´ \_connect\_identifier> "
  - SYS :AS SYSDBA on 12/19 14:11 using DB01>
- ### connect\_identifier 확인 : select \* from global\_name;
- Global 설정 파일 : \$ORACLE\_HOME/sqlplus/admin/glogin.sql
  - 예를 들어서 glogin.sql 파일을 열어서 다음과 같이 셋팅한 후 저장하도록 한다
  - set sqlprompt "\_user> "

# SQL\*Plus 접속/종료

보류

사용자	접속 종료 위치	실행 방법
관리자	운영 체제	\$ sqlplus / as sysdba
	SQL Prompt	SQL > conn[ect] / as sysdba
	SQL Prompt	SQL > Exit (완전 종료)
	SQL Prompt	SQL > disconnect (접속 종료)
일반 사용자	운영 체제	\$ sqlplus 아이디/비밀번호
	SQL Prompt	SQL > conn[ect] 아이디/비밀번호
	SQL Prompt	SQL > Exit (완전 종료)
	SQL Prompt	SQL > disconnect (접속 종료)

# Oracle(SQL)

테이블(Table)



# 객체 명명 규칙

카페

- Oracle에서 객체 생성시 이름을 작성하는 규칙이 있다.

항목	설명
객체(Object)	Oracle DB를 구성하고 있는 요소들(테이블, 시퀀스, 인덱스 등등)
작성 규칙	반드시 문자로 시작한다. a123(O), 3abc(X)
특징	길이는 1 ~ 30. (단, DB_Name은 8바이트 이내.) 알파벳 대소문자 및 숫자, 특수 기호 일부(underscore, \$, #)가 사용될 수 있다. 컬럼 이름의 끝에 #이 있으면 숫자형 컬럼이다. \$ : 동적 성능 뷰에 사용.(v\$XXX 뷰) 동일 사용자의 다른 객체 이름과 중복이 불가능하다. Oracle에서 사용하는 예약어는 사용 불가능하다. 사용되고 있는 예약어는 다음 문장을 이용하여 확인할 수 있다. conn / as sysdba select * from v\$reserved_words;

# Data Type

보류

- 필드(컬럼)를 사용할 수 있는 데이터 타입은 다음과 같은 항목들이 존재한다.

이름	비고
CHAR (size)	고정 길이 문자, 입력 자료와 상관 없이 길이만큼 공간 차지 1<= 크기 <=2000
VARCHAR2(size)	가변 길이 문자, 실제 입력된 데이터 만큼 공간 차지, 1<= 크기 <=4000
NUMBER	최대 40자리의 숫자를 저장 가능.
NUMBER(w)	w 자리까지의 수치로 38자리까지 가능.
NUMBER(w, d)	w는 전체 길이, d는 소수점 이하 자리수.
DATE	BC 4712년 1월 1일 ~ AD 4712년 12월 31일
LONG	가변 길이의 문자형 데이터 타입, 최대 크기는 2GB
CLOB	대형 문서 파일, 연설문, 4GB까지 문자열 데이터.
BLOB	이미지, 동영상, 4GB까지 바이너리 데이터.
ROWID	테이블의 ROW에 붙여 있는 고유 식별 주소, 인덱스에 사용됨.



# 테이블 생성

사용 형식(신규 테이블 생성)

```
CREATE TABLE 테이블_이름(  
    컬럼_이름 데이터_타입 [DEFAULT 기본값],  
    컬럼_이름 데이터_타입 [DEFAULT 기본값]  
    ...  
)
```

사용 예시

```
create table employees(  
    sabun number(6),  
    name varchar2(20),  
    email varchar2(25),  
    phone varchar2(20),  
    hiredate date default sysdate,  
    salary number(8,2),  
    managerid number(6)  
) ;
```

조회

```
-- 내가 소유한 테이블 리스트 보기  
select * from tab;  
  
-- books 테이블 구조 간략하게 보기  
desc books  
  
-- books 테이블 내용물 보기  
select * from books ;
```

# 테이블 생성

## 사용 형식(CTAS 방식)

```
CREATE TABLE 테이블_이름  
AS  
select * | 컬럼  
FROM 이전테이블이름
```

## 테이블 복제를 하게 되면?

- ① Structure(Column) 복사됨.
- ② Data(Row)도 복사됨
- ③ Not Null 제약 조건은 복사됨
- ④ 나머지 제약 조건은 복사되지 않는다.

## 사용 예시

employees 테이블의 사번/이름/입사일자만  
복제하여 테이블 emp2를 생성하세요.

```
create table emp2  
as  
select sabun, name, hiredate  
from employees;
```

## 조회

```
select * from tab;  
select table_name  
from user_tables  
order by table_name ;
```

```
desc emp2  
select * from emp2 ;
```

# 테이블 생성

카페

## 사용 형식(테이블 복제)

```
CREATE TABLE 테이블_이름(컬럼이름1, 컬럼이름2, ...)  
AS  
select *|컬럼  
FROM 이전테이블이름;
```

참고 : 테이블 구조만 복사하고자 하는 경우에는 where 절에 where 1 = 2을 사용한다.

## 사용 예시

employees 테이블의 사번/이름/입사일자만  
복제하여 테이블 emp3를 생성하세요.  
단, 컬럼명은 id/ename/hdate으로 생성하시오.

```
create table emp3  
as  
select sabun, name, hiredate  
from employees;
```

## 조회

```
select * from tab;  
desc emp3  
select * from emp3 ;
```

# DEFAULT 옵션

- 디폴트(Default)

- 사용자가 직접 입력하지 않았을 경우에 자동으로 기입되는 초기 값을 의미한다.
- 컬럼\_이름 데이터\_타입 [DEFAULT 기본값])

## 사용 예시

```
create table emp4(  
    empno number,  
    name varchar2(15) default '호호',  
    sal number(7, 2) default 300  
);  
-- 명시적 디폴트를 추가하는 예시 : 사용자가 Default라는 키워드를 사용하면 된다.  
insert into emp4 values(1, default, default) ;  
-- 암시적 디폴트를 추가하는 예시 : 시스템이 알아서 생성을 해준다.  
insert into emp4(empno) values(2) ;  
-- 사용자가 어떠한 값을 넣게 되면 Default는 무시된다.  
insert into emp4 values(3, '김철수', 999) ;  
commit ;  
select * from emp4;  
drop table emp4 purge ; -- 테이블 영구 삭제하기
```

# colview.sql 파일 만들기

보류

- 데이터 사전 (Data Dictionary)
  - 데이터베이스 객체를 관리하기 위하여 사용하는 내장 (built-in) 사전
- user\_tab\_columns 데이터 사전
  - 사용자의 특정 테이블에 대한 컬럼 정보를 보여주는 데이터 사전이다
  - 컬럼 이름 / 데이터 종류/ 길이 / 기본 값 여부 등등

## 사용 예시

```
--테이블의 컬럼 정보 보기
col col_name for a15
col type for a10
col default for a10

select column_name col_name,
data_type "type",
data_length "length",
data_default "default", nullable
from user_tab_columns
where table_name=upper('&table_name') ;
```

## 파일로 저장

```
--자주 사용하는 문장이므로 파일로 저장하기
SQL> save c:\labs\colview.sql 엔터

SQL> @c:\labs\colview 엔터
table_name 입력 : emp2 엔터

COL_NAME type length default NU
-----
SABUN NUMBER 22 Y
NAME VARCHAR2 20 Y
HIREDATE DATE 7 Y
```

# 테이블 수정( 컬럼 추가 )

카페

- 이전 테이블에 새로운 컬럼 또는 제약 조건을 추가한다
- 이미 존재하는 행이 있다면 , 모든 행에 대하여 Null 으로 초기화된다

## 사용 형식

```
ALTER TABLE 테이블_이름 ADD ( 컬럼_이름 데이터_타입 );
```

## 사용 예시

```
DESC employees
```

```
--employees 테이블에 Job 컬럼 추가  
ALTER TABLE employees  
ADD (JOB VARCHAR2(9));
```

```
--컬럼 2개이상 추가시 [,] 으로 연결  
ALTER TABLE employees  
ADD (COL1 NUMBER,  
COL2 DATE DEFAULT SYSDATE);
```

```
DESC employees
```

## 조회

```
--emp2 테이블에 job 컬럼을 추가하되  
--크기는 varchar2(10) 으로 설정하시오.
```

```
--colview.sql 파일을 이용하여 테이블의  
--컬럼 정보를 조회하세요
```

```
@colview.sql
```

# 테이블 수정(컬럼 수정)

- 컬럼에 대한 type, 길이, default 값을 변경한다
- 컬럼의 default 값은 이후에 추가되는 행에 대해서만 영향을 미친다

## 사용 형식

```
ALTER TABLE 테이블_이름 MODIFY ( 컬럼_이름 데이터_타입 );
```

종류	설명
해당 컬럼에 자료가 없는 경우	컬럼의 데이터 타입을 변경할 수 있다. 컬럼의 크기를 변경할 수 있다
해당 컬럼에 자료가 있는 경우	컬럼의 데이터 타입을 변경할 수 없다. 크기를 늘릴 수는 있지만 , 데이터 크기보다 작은 크기로 변경 불가능

## 사용 예시

```
desc employees
```

```
alter table employees  
modify (job varchar2(30));
```

```
desc employees
```

## 조회

```
@colview.sql
```

# 테이블 수정( 컬럼 이름 변경/삭제 )

카페

- 컬럼 이름을 변경한다.

사용 형식

```
alter table 테이블_이름 rename column  
이전_컬럼_이름 to 새로운_컬럼_이름 ;
```

사용 예시

```
alter table employees  
rename column job  
to love;
```

```
--employees 테이블의 col1, col2 를 test1,  
test2 으로  
--변경해보세요
```

```
desc emp01;
```

```
@colview.sql
```

- 컬럼을 삭제하는 데 데이터가 들어 있어도 컬럼 삭제가 가능하다
- 한번에 1 개의 컬럼만 삭제가 가능하다 ( 복구 불가능 )

사용 형식

```
alter table 테이블_이름  
drop column ( 컬럼_이름 );
```

사용 예시

```
desc employees
```

```
alter table employees drop column test1 ;
```

```
alter table employees drop column love;
```

```
@colview.sql
```



# 테이블 삭제

카페

- 테이블을 삭제한다.

## 사용 형식

```
drop table 테이블_이름 [purge] ;  
purge 옵션은 영구 삭제를 사용
```

## 사용 예시

```
drop table employees ;
```

```
select * from tab ;
```

```
tname tabtype clusterid
```

```
-----  
중략BIN$fB3ZG6E7Q02lfMwElbVRgg==$0 TABLE
```

```
--휴지통에 있는 employees 테이블 복구하기  
flashback table employees to before drop;
```

```
select * from tab;  
desc employees  
select * from employees ;
```

# 기타 DDL

카페

종류	설명
RENAME	테이블 이름을 변경한다
TRUNCATE	테이블의 모든 행을 삭제한다

사용 형식
RENAME 이전_테이블_이름 TO 신규_테이블_이름 ;

사용 예시
RENAME EMP2 TO EMP99;  TRUNCATE TABLE EMP3;

# set unused 옵션

카페 2345

- 자주 사용되지 않거나 필요없다고 판단이 되는 컬럼은, 삭제를 해야 한다
- 업무 시간에 삭제를 수행하게 되면 소요되는 시간 및 다른 사용자의 업무에 방해를 줄 수 있다.  
(undo 데이터의 생성으로 인한 )
- 다른 사용자가 이 컬럼에 대하여 접근을 하지 못하게 막으려면 set unused 옵션을 사용하면 된다.
- set unused 컬럼으로 지정하게 되면 접근이 불가능하게 되지만, 물리적인 공간은 차지한다.
- drop unused columns 으로 삭제가 가능하다
- 설정된 컬럼은 다시 활성화 시킬 수 없으므로 신중하게 고려 해야 한다.
- unused column 확인
  - USER\_UNUSED\_COL\_TABS ( table\_name, count )

## 사용 형식

```
ALTER TABLE 테이블_이름 set unused ( 컬럼_이름 );  
ALTER TABLE 테이블_이름 drop unused columns ;
```

## 사용 예시

```
DESC EMP3  
ALTER TABLE EMP3 SET UNUSED (hiredate );  
DESC EMP3  
select * from EMP3;
```

# set unused 옵션 예시

카페 2345

- desc emp03
- 이름 -널? 유형
- -----
- SABUN NUMBER(6)
- NAME VARCHAR2(20)
- HIREDATE DATE
- --HIREDATE 컬럼을 더 이상 사용을 하지 못하게 차단하려면
- ALTER TABLE EMP03 SET UNUSED (HIREDATE);
- desc emp03
- 이름 널? 유형
- -----
- SABUN NUMBER(6)
- NAME VARCHAR2(20)
- select table\_name, count from user\_unused\_col\_tabs;
- --EMP03 테이블에 set unused 옵션이 걸린 컬럼이 1 개이다
- TABLE\_NAME COUNT
- -----
- EMP03 1
- --직원들 모두 퇴근 한 시간에 다음 작업을 수행 바람
- alter table emp03 drop unused columns ;
- select table\_name, count from user\_unused\_col\_tabs;
- 선택된 레코드가 없습니다 <--unused 옵션 적용된 칼럼이 없다는 소리

# Oracle(SQL)

데이터 조작어(DML)



# 데이터 조작용어(DML)

카페

- 테이블의 행(row)에 대하여 추가/수정/삭제 등을 수행하기 위한 언어
- 관련 지식
  - 트랜잭션(transaction)
  - undo 데이터에 대한 지식

기능	비고
insert	테이블에 새로운 행을 추가한다.
update	테이블의 내용을 수정한다.
delete	테이블의 특정 행을 삭제한다.

# 인서트(insert)

카페

- 행(row)을 추가하고자 할 때 insert 구문을 사용한다.

항목	설명
내용	해당 테이블에 행(row)을 추가한다.
사용 문법	insert into 테이블_이름 ( 컬럼 01, 컬럼 02, ..., 컬럼 nn) values( 입력값 01, 입력값 02, ..., 입력값 nn);
특징	한번에 하나의 행만 삽입된다. 컬럼 목록을 지정하지 않으면 테이블 생성시 만들어진 컬럼의 순서대로 데이터가 추가된다. 기존된 컬럼 순서대로 values에 지정된 값이 추가된다.
주의 사항	열거하는 값의 개수 및 데이터 타입은 반드시 동일해야 한다 문자와 날짜 값은 외따옴표(' ') 으로 둘러 싸야 한다 최종 작성 후에 반드시 커밋(commit) 을 수행하도록 한다 만약 수행 결과를 취소하려면 롤백(rollback) 을 수행하도록 한다.

# 인서트 예시

카페

- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(1, '이수만', 'x@aacom', '01035119875', sysdate, 100, null) ;
- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(2, '양현석', 'x@aacom', '01035119875', sysdate, 100, null) ;
- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(3, '박진영', 'x@aacom', '01035119875', sysdate, 100, null) ;
- -----
- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(4, '제시카', 'x@aacom', '01035119875', sysdate, 100, 1) ;
- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(5, '티파니', 'x@aacom', '01035119875', sysdate, 100, 1) ;
- insert into employees(sabun, name, email, phone, hiredate, salary, managerid)
- values(6, '수영', 'x@aacom', '01035119875', sysdate, 100, 1) ;
- -----
- col name for a10
- col email for a12
- col phone for a12
- commit ;
- select \*from employees ;



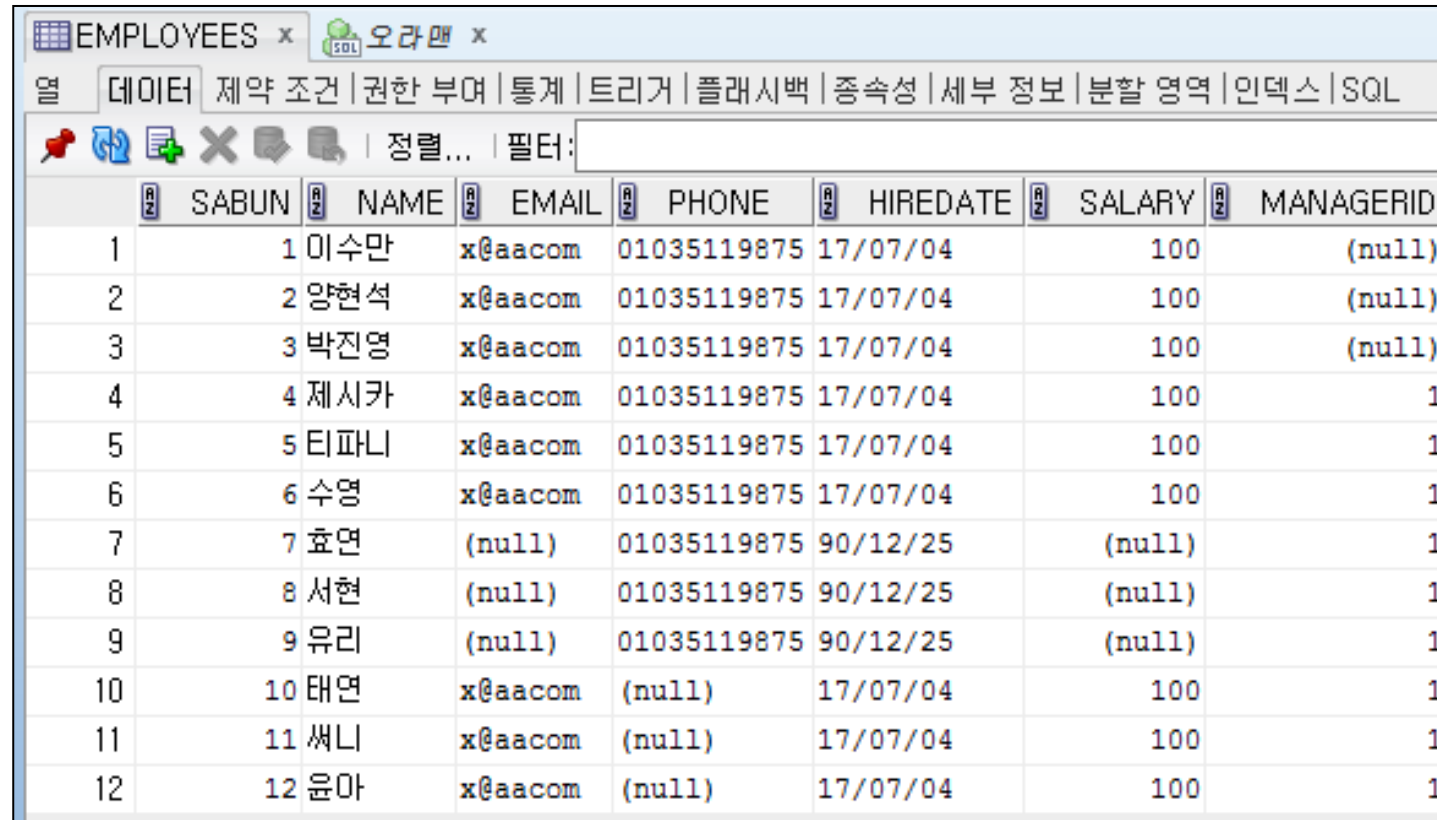
# 인서트 예시

카페

- insert into employees(sabun, name, phone, hiredate, managerid)
- values(7, '호연', '01035119875', to\_date('90/12/25'), 1) ;
- insert into employees(sabun, name, phone, hiredate, managerid)
- values(8, '서현', '01035119875', to\_date('90/12/25'), 1);
- insert into employees(sabun, name, phone, hiredate, managerid)
- values(9, '유리', '01035119875', to\_date('90/12/25'), 1) ;
- -----
- -- 컬럼의 순서는 바뀌어도 상관 없다
- insert into employees(salary, sabun, name, email, hiredate, managerid)
- values(100, 10, '태연', 'x@aacom', sysdate, 1) ;
- insert into employees(salary, sabun, name, email, hiredate, managerid)
- values(100, 11, '씨니', 'x@aacom', sysdate, 1) ;
- insert into employees(salary, sabun, name, email, hiredate, managerid)
- values(100, 12, '윤아', 'x@aacom', sysdate, 1) ;
- 
- commit ;
- 
- -- 사번 순으로 오름차순 정렬하세요.
- select \*from employees order by sabun ;

# 인서트 예시

- 현재까지 테이블에 들어간 데이터 목록이다.



The screenshot shows an Oracle SQL Developer window with the 'EMPLOYEES' table selected. The window has a menu bar with options like '데이터', '제약 조건', '권한 부여', etc. Below the menu is a toolbar with icons for saving, undo, redo, and others. The main area displays a table with 12 rows of employee data. The columns are SABUN, NAME, EMAIL, PHONE, HIREDATE, SALARY, and MANAGERID. The data is as follows:

	SABUN	NAME	EMAIL	PHONE	HIREDATE	SALARY	MANAGERID
1	1	이수만	x@aacom	01035119875	17/07/04	100	(null)
2	2	양현석	x@aacom	01035119875	17/07/04	100	(null)
3	3	박진영	x@aacom	01035119875	17/07/04	100	(null)
4	4	제시카	x@aacom	01035119875	17/07/04	100	1
5	5	티파니	x@aacom	01035119875	17/07/04	100	1
6	6	수영	x@aacom	01035119875	17/07/04	100	1
7	7	효연	(null)	01035119875	90/12/25	(null)	1
8	8	서현	(null)	01035119875	90/12/25	(null)	1
9	9	유리	(null)	01035119875	90/12/25	(null)	1
10	10	태연	x@aacom	(null)	17/07/04	100	1
11	11	써니	x@aacom	(null)	17/07/04	100	1
12	12	윤아	x@aacom	(null)	17/07/04	100	1

# 인서트 예시

- 다음의 데이터들을 모두 인서트하고 커밋 하도록 한다.

SABUN	NAME	EMAIL	PHONE	HIREDATE	SALARY	MANAGERID
13	탐	x@aa.com	01035119875	13/06/28	100	2
14	지용	x@aa.com	01035119875	13/06/29	120	2
15	대성	x@aa.com	01035119875	13/06/30	140	2
16	승리	x@aa.com	01035119875	13/06/31	160	2
17	태양	x@aa.com	01035119875	13/06/32	180	2
18	선미	y@bb.com	01035119875	13/06/32	200	3
19	선예	y@bb.com	01035119875	13/06/32	220	3
20	소희	y@bb.com	01035119875	13/06/32	240	3
21	유빈	y@bb.com	01035119875	13/06/32	260	3
22	예은	y@bb.com	01035119875	13/06/32	280	3

# 업데이트(update)

카페

- 행(row)을 수정하고자 할 때에는 update 구문을 사용한다.

항목	설명
내용	이미 저장되어 있는 행에 대한 데이터를 수정하기 위한 방법이다.
사용 문법	update 테이블_이름 SET 컬럼1 = 값1, 컬럼2 = 값2, ... 컬럼n = 값n [ where 조건식 ];
주의 사항	문자와 날짜 값은 외따옴표 ( ' ' ) 으로 둘러 싸야 한다. where 절을 지정하지 않으면 모든 행이 수정된다.

# 업데이트 예시

보류

내용	sql 구문
모든 직원들의 급여를 100 으로 설정하라	update employees set salary=100 ;
10 번 이상의 직원들의 급여를 300 으로 설정하라	update employees set salary=300 where sabun >= 10;
위의 사항들을 롤백하시오.	rollback ;
위의 사항들을 다시 실행하시오.	상단의 실행 다시 수행
모든 직원들의 급여를 10% 인상 시켜 보자.	update employees set salary=1.1*salary ;
모든 직원의 입사일을 오늘로 수정해보자.	update employees set hiredate=sysdate;
매니저 id 가 1인 직원들의 급여를 50 으로 설정하라.	update employees set salary=50 where managerid = 1;
매니저 id 가 2 인 직원들의 급여를 현재 사번의 곱하기 100 으로 설정하라.	update employees set salary=100*sabun where managerid = 2;
사번이 짝수인 직원들은 급여를 300으로, 입사일자를 3 개월 전으로 설정하라.	update employees set salary=300, hiredate=add_months(sysdate, -3) where mod(sabun, 2) = 0;
사번이 홀수인 직원들은 급여를 500으로, 입사일자를 5 개월 전으로 설정하라.	update employees set salary=500, hiredate=add_months(sysdate, -5) where mod(sabun, 2) = 1;
사번 1 의 직원명 ( 홍길동 ), 급여 (1200), 이메일 (dd @abccom) 으로 변경하시오.	update employees set name=' 홍길동 ', salary=1200, email='dd@abccom' where sabun= 1;
위의 사항들을 영구 저장 (commit) 하시오.	commit ;

# 딜리트(delete)

카페

- 행(row)을 삭제하고자 할 때에는 delete 구문을 사용한다.

항목	설명
내용	테이블의 특정 행을 삭제한다.
사용 형식	delete from 테이블_이름 [ where 조건식 ];
사용 예시	delete from students where id = 'hong'
주의 사항	문자와 날짜 값은 외따옴표 ( ' ' ) 으로 둘러싸야 한다. where 절을 지정하지 않으면 모든 행이 삭제된다. 실제 내용인 데이터만 삭제되고 , 테이블 자체는 삭제되지 않는다. delete 명령어로 column 삭제는 불가능하다. 컬럼의 삭제는 alter table 명령어를 사용해야 한다.

# 딜리트 예시

보류

내용	sql 구문
사원 4 를 삭제하시오.	delete from employees where sabun = 4 ;
매니저 id 가 2 이고, 사번이 짝수인 사원을 삭제하라.	delete from employees where managerid = 2 and mod(sabun, 2) = 0 ;
매니저 id 가 3 이거나, 사번이 홀수인 사원을 삭제하라.	delete from employees where managerid = 3 or mod(sabun, 2) = 1 ;
위의 사항들을 롤백하시오.	rollback ;
모든 사원들의 급여 = 100 * 사번으로 설정하라.	update employees set salary=100*sabun ;
위의 사항들을 영구 저장 (commit) 하시오.	commit ;
매니저 id 가 1 이고, 급여가 600 이하 또는 1000 이상인 행을 삭제하라.	delete from employees where managerid=1 and (salary<=600 or salary >= 1000 ) ;
급여가 1300 이상이고, 1700 이하인 행을 삭제하라.	delete from employees where salary between 1300 and 1700 ;
매니저 id 가 없는 사원들을 삭제하시오.	delete from employees where managerid is null ;
모든 사원을 삭제하시오.	delete from employees ;
위의 사항들을 영구 저장 (commit) 하시오.	commit ;

DML 실습(문제).pdf

# Oracle(SQL)

시퀀스(Sequence)





# 시퀀스(Sequence)

카페

- 유일(unique)한 정수 값을 생성해주는 Oracle 객체이다.

항목	설명
특징	순차적으로 증가 /감소하는 값을 가질 수 있다. 특정 테이블의 단독 소유물이 아닌, 여러 객체 간에 공유하는 객체이다. 통상적으로 테이블 1개당 시퀀스 1개를 묶어서 사용한다.
용도	기본 키 : 테이블 내의 행을 식별하기 위한 유일한 컬럼 테이블의 기본 키를 자동으로 입력하기 위한 수단으로 사용된다 no cycle 옵션을 사용해야 한다.
데이터 사전	USER_SEQUENCES 디렉터리 뷰 현재 소유하고 있는 시퀀스 리스트 보기 select * from seq ; select sequence_name from user_sequences;

# 시퀀스 사용 형식

카페

## 사용 형식

```
CREATE SEQUENCE 시퀀스이름  
START WITH n  
INCREMENT BY n  
[MAXVALUE n|NOMAXVALUE]  
[MINVALUE n|NOMINVALUE]  
[CYCLE|NOCYCLE]  
[CACHE n|NOCACHE]
```

문법	설명
START WITH	최초 시작 값
INCREMENT BY	증가/감소할 값
MAXVALUE n   NOMAXVALUE	증가할 수 있는 최대 값
MINVALUE n   NOMINVALUE	시퀀스의 최소 값, 기본 값은 1
CYCLE   NOCYCLE	값의 마지막에 도달시 다시 처음부터 시작하는 사이클 옵션. primary key로 사용하는 경우 nocycle 옵션을 사용해야 한다.
CACHE n   NOCACHE	메모리에 미리 로딩할 개수

# CURRVAL/NEXTVAL

카페

- CURRVAL
  - 현재의 시퀀스 값을 알아낼 수 있다.(current value)
- NEXTVAL
  - 다음 시퀀스 번호를 구할 수 있다.(next value)

문법	설명
NEXTVAL/CURRVAL 사용 가능한 경우	서브 쿼리가 아닌 select 문 insert 문의 select 절 insert 문의 VALUE 절 update 문의 SET 절
NEXTVAL/CURRVAL 사용 불가능한 경우	VIEW의 select 절 DISTINCT 가 키워드가 있는 select 문 GROUP BY, HAVING, ORDER BY 절이 있는 select 문 select, delete, update의 서브 쿼리 CREATE TABLE, ALTER TABLE 명령의 DEFAULT 값

# 시퀀스 생성 실습

카페

- 시작 값 =1, 증가 값 =1, 최대값 =100000 시퀀스 EMP\_SEQ 를 생성하시오

## 사용 예시

```
create sequence emp_seq start with 1 increment by 1 maxvalue 100000 ;  
select * from user_sequences;
```

```
--시퀀스 실습을 위한 테이블 생성  
delete from employees ;  
commit ;
```

```
--시퀀스 이용한 데이터 INSERT  
insert into employees(sabun, name, email, phone, hiredate, salary, managerid)  
values( emp_seq.nextval , ' 이수만 ', 'x @aacom', '01035119875', sysdate, 100,  
null) ;  
insert into employees(sabun, name, email, phone, hiredate, salary, managerid)  
values( emp_seq.nextval, ' 박진영 ', 'x @aacom', '01035119875', sysdate, 100, null) ;  
insert into employees(sabun, name, email, phone, hiredate, salary, managerid)  
values( emp_seq.nextval, ' 양현석', 'x@aacom', '01035119875', sysdate, 100, null) ;
```

시퀀스를 이용하여 이전에 실습했던 모든 데이터를 다시 INSERT 하시오

# 시퀀스 생성 실습

카페

- 시퀀스가 최대 값을 초과하면 어떻게 되는 가 ?
- 시작 값 =10, 증가 값 =10, 최대값 =30 시퀀스 dept\_seq 을 생성

## 사용 예시

```
create sequence dept_seq start with 10 increment by 10 maxvalue 30;
```

```
create table dept01(id number, name varchar2(30), location varchar2(30)) ;
```

```
insert into dept01 values(dept_seq.nextval, 'research', 'dalass');
```

```
insert into dept01 values(dept_seq.nextval, 'sales', 'chicago');
```

```
insert into dept01 values(dept_seq.nextval, 'operations', 'boston');
```

--아래는 오류가 발생한다.( 최대 값을 초과 )

```
insert into dept01 values(dept_seq.nextval, 'purchase', 'seoul');
```

2행에 오류:

ora-08004: 시퀀스 dept\_seq.nextval exceeds

maxvalue 은 사례로 될수 없습니다

# 시퀀스 생성 실습

카페

- USER\_SEQUENCES 데이터 사전 조회

## 사용 예시

```
select sequence_name, min_value,  
max_value, increment_by, cycle_flag  
from user_sequences;
```

--시퀀스 값의 초과로 의하여 인서트 불가  
--시퀀스의 최대 값 변경하기

```
alter sequence dept_seq  
maxvalue 100000;
```

alter sequence에서  
start with 절을 제외한 나머지 구문은  
create sequence처럼 모두 사용이 가능하다.

## 사용 예시

```
insert into dept01  
values(dept_seq.nextval, 'operations', 'boston');
```

```
select * from dept01;
```

--시퀀스의 삭제

```
drop sequence dept_seq;
```

# Oracle(SQL)

트랜잭션(Transaction)



# 트랜잭션(Transaction)

카페

항목	설명
개념	데이터를 처리하는 하나의 논리적인 처리 단위
목적	일관성 유지(데이터 무결성이 보장된다) 안정적인 데이터를 복구하기 위함
특징	ALL-OR-NOTHING 방식 논리적으로 연관된 작업을 그룹화할 수 있다.
시작과 종료	트랜잭션 시작 : 최초 DML 문장(DML이 실행됨과 동시에 트랜잭션이 시작) 끝 : commit(영구 저장) 또는 rollback(이전 작업 취소) commit과 rollback은 상호 배타적이다 DML 문장은 반드시 커밋 또는 롤백을 하여 트랜잭션 종료를 해줘야 한다

명령어	의미
commit	트랜잭션의 처리 과정을 데이터베이스에 영구 저장 이전 데이터가 완전히 업데이트됨 모든 사용자가 변경된 데이터의 결과를 볼 수 있다.
rollback	발생한 변경 사항을 취소하는 명령어 이전 COMMIT 한 곳 까지 복구가 됨



# SAVEPOINT

보류

- 현재의 트랜잭션을 작게 분할하는 기능이다.
- ROLLBACK TO SAVEPOINT 문을 사용하여 표시한 곳까지 ROLLBACK 가능

## 사용 형식

-- SAVEPOINT로 특정 위치를 지정하기 위하여 사용하는 형식  
savapoint 레이블\_이름 ;

-- SAVEPOINT 로 특정 위치로 돌아가기 위한 명령어  
ROLLBACK TO 레이블\_이름 ;

# Oracle(SQL)

기본 select 구문



# employees 테이블 예시

보류

SABUN	NAME	EMAIL	PHONE	HIREDATE	SALARY	MANAGERID
1	미수만	x@sa.com	01035119875	13/06/29	100	-
2	박진영	x@sa.com	01035119875	13/06/29	100	-
3	양현석	x@sa.com	01035119875	13/06/29	100	-
4	제시카	x@sa.com	01035119875	13/06/29	100	1
5	티파니	x@sa.com	01035119875	13/06/29	100	1
6	수영	x@sa.com	01035119875	13/06/29	100	1
7	효연	-	01035119875	90/12/25	-	1
8	서현	-	01035119875	90/12/25	-	1
9	유리	-	01035119875	90/12/25	-	1
10	유리	x@sa.com	- - - - -	13/06/29	100	1
11	씨니	x@sa.com	-	13/06/29	100	1
12	윤아	x@sa.com	-	13/06/29	100	1
13	탈	x@sa.com	01035119875	13/06/28	100	2
14	지용	x@sa.com	01035119875	13/06/28	100	2
15	대성	x@sa.com	01035119875	13/06/28	100	2
16	승리	x@sa.com	01035119875	13/05/28	100	2
17	태양	x@sa.com	01035119875	13/06/28	100	2
18	선미	x@sa.com	01035119875	13/06/28	100	3
19	선예	x@sa.com	01035119875	13/06/28	100	3
20	소희	x@sa.com	01035119875	13/06/28	100	3
21	유빈	x@sa.com	01035119875	13/06/28	100	3
22	예은	x@sa.com	01035119875	13/06/28	100	3

# SQL 구문 작성 지침

- 키워드 예약어 등에 대한 대소문자 구분이 없다
- 여러 라인에 걸쳐서 작성 가능하다.
- 키워드는 줄여서 쓰거나 나뉘어 쓰지 마라.
- 절과 절은 나뉘어 쓰라.
- 적절한 들여 쓰기는 가독성을 향상시킨다.
- 세미콜론 (;) 을 사용하여 문장의 끝을 표시하라.
- 세미콜론(;) 대신 / 으로도 사용이 가능하다.(/는 단독 라인에서 사용 가능)

# select 기본 문형

카페

- 데이터를 조회하는 select 구문의 기본 문형은 다음과 같다.

## select 기본 문법

```
select *|[[distinct] columnlexpression [alias],]  
from 테이블 1[, 테이블 2, ...]  
[where condition]  
[group by column]  
[having column]  
[order by column];
```

SQL 문장	항목에 대한 설명
select	한 개 또는 그 이상의 컬럼 이름을 열거하는 절(clause)이다.
*	모든 컬럼(all columns)을 의미한다.
distinct	중복된 값 배제.(중복된 데이터는 1건으로 처리한다.)
column   Expression	컬럼 이름 또는 표현식( 컬럼의 산술 연산화 )
alias(알리아스)	컬럼의 별칭, 다른 이름으로 표시하기
from Table	컬럼을 담고 있는 테이블을 지정한다.

# 컬럼 조회하기

- 테이블 구조 보기 : DESC 테이블이름

내용	sql 구문
모든 사원들의 정보를 조회하시오.	select * from employees order by sabun ;
모든 사원들의 이름/메일/전화 번호 조회.	select name, email, phone from employees order by sabun ;
모든 사원들의 매니저 아이디를 조회.	select managerid from employees order by sabun ;
모든 사원들의 매니저 아이디를 조회. 단, 중복된 행은 배제하도록 하세요.	select distinct managerid from employees ;
모든 사원들의 이름/급여/연봉 조회. 단, 연봉은 급여의 12배수이다.	select name, salary, 12*salary from employees order by sabun ;
* 는 +보다 우선 순위가 높다.	select name, salary, 12*salary + 100 from employees order by sabun ;
( ) 에 의한 연산 순서의 변경	select name, salary, 12*(salary + 100) from employees order by sabun ;

# Alias( 별칭 ) 사용

카페

- 컬럼 이름을 사용자가 알아 보기 쉬운(?) 이름으로 재정의한다.
- 계산식에 유용하게 사용할 수 있다.
- AS 키워드는 옵션(Optional) 사항이다.
- 스페이스, 특수 문자, 대소문자 유지는 쌍따옴표("")로 처리한다.
- 파생된 컬럼
  - 실제 존재하지 않고, 어떤 계산식에 의하여 발생한 컬럼에서 사용될 수 있다.
  - JDBC 프로그램에서 사용시 파생된 컬럼은 반드시 alias를 만들어 주도록 한다.

내용	sql 구문
Alias 를 사용하여 모든 직원들의 이름 / 급여 /연봉을 조회하시오. 단 , 연봉은 급여의 12배수이다.	select name, salary,12*salary as 연봉 from employees order by sabun ;
모든 직원들의 이름/메일/전화 번호 조회. 단 , Alias 를 사용하세요.	select name as 이름 , email 이메일 , phone "전화 번호" from employees order by sabun ;

# Null

카페

- 비교 판단이 불가능한 정의할 수 없는 어떠한 값을 의미한다.
  - $5 > 7$  (true)
  - $null > 7$  (알 수 없음)
  - $null > null$  (알 수 없음)
- 값이 없는 것  $\neq$  null 값
- 특징
  - 연산 결과 값은 모두 Null 이 됨
  - IS NULL으로 비교 가능하다
  - 연결 연산자와 합쳐지는 경우에는 흡수된다

내용	sql 구문
이메일 주소가 null 직원들의 이름/급여 / 연봉을 조회하시오.	<code>select name, salary, 12*salary as 연봉 from employees where email is null order by sabun ;</code>
이메일 주소가 null이거나 phone이 null 인 직원들의 이름/급여/이메일/전화번호를 조회하시오.	<code>select name, salary, email, phone from employees where email is null or phone is null order by sabun ;</code>



# 연결 연산자

카페

- 컬럼을 다른 문자열 또는 컬럼과 연결하기 위한 용도로 사용한다.
- 연산 결과는 최종적으로 문자열이 된다.
- 2개의 수직 바로 구성한다(II)

내용	sql 구문
홍길동의 급여는 100 원입니다.	select name   '의 급여는 '  salary   ' 원입니다 ' from employees ;
이름 : 홍길동 , 이메일 : xxx@navercom	select ' 이름 : '  name  ', 이메일 : '  email from employees ;
홍길동의 이메일 주소는 xxx@navercom입니다.	select name    ' 의 이메일 주소는 '    email    ' 입니다 ' from employees ;

# Quote 연산자 (Q 연산자 )

- 표시 불가능한 문자 ( 예 : ' 문자 ) 를 결과에 보여 주고자 하는 경우에 사용
- ' Mark구분 문자 (Delimiter)을 선택하라
- 구분 문자 (Delimiter) 을 선택하라 ([ ]/ { }/ ( )/ < > 중 택1)
- 표시 형식 : q'X 표시할 내용 X '
  - q 로 시작하고 '으로 둘러싼다.
  - X 는 임의의 1 자리 문자를 말한다.
- 가독성과 사용성을 증가시킨다

## 사용 예시

```
select department_name ||q'[, it's assigned Manager Id:]'  
||manager_id AS "Department And Manager"  
from departments ;  
Department And Manager  
-----  
Administration, it's assigned Manager Id:200  
Marketing, it's assigned Manager Id:201  
Purchasing, it's assigned Manager Id:114
```

Q 연산자 txt

# Oracle(SQL)

행의 제한과 정렬



# 정렬과 행의 제한

카페

## select 기본 문법

```
select *|{[distinct] columnlexpression [alias],}  
from table  
where 조건식  
order by 정렬_방식;
```

조건식	설명
컬럼 이름	조건식에는 컬럼 이름을 열거할 수 있다.
수식/표현식	컬럼에 대한 수식 및 표현식이 올 수 있다.
상수	일정한 값을 가지는 상수 가능.

정렬 방식	설명
기본 값	오름차순이 기본 값(ASC ↔ DESC)
위치	order by 절은 select 구문의 마지막에 위치한다
특이 사항	해당 컬럼이 select 리스트 목록에 존재하지 않더라도 order by 절에 컬럼을 표시할 수 있다.

# and/or/not 연산자

카페

연산자	의미
AND	논리곱([그리고] 라는 논리적 언어로 연결된 합성 명제)
OR	논리합([또는] 이라는 논리적 언어로 연결된 합성 명제)
NOT	부정(진위 값을 변경하는 것)

OR 진위표	true	false	null
결과	false	true	null

AND 진위표	true	false	null
true	true	false	null
false	false	false	false
null	null	false	null

OR 진위표	true	false	null
true	true	true	true
false	true	false	null
null	true	null	null

## not 연산자의 위치

```
where job_id not IN ('AC_ACCOUNT', 'AD_VP')
where salary not between 10000 AND 15000
where last_name not like '%A%'
where commission_pct IS not null
```

# 비교 연산자

카페

연산자	의미
=	같다.
>	크다.
>=	크거나 같다.
<	작다.
<=	작거나 같다.
<>	같지 않다.
mod(a, b)	나머지 연산자, a를 b로 나눈 나머지
between ... and ...	between two values (inclusive) 양쪽 값을 포함한다.(Range Value)
in( set )	셋트 목록 중의 하나와 일치, OR 연산자의 조합
like	문자 패턴과 일치(반드시_또는 %와 같이 사용)
is null	null 데이터가 들어 있는 항목만 조회 ↔ is not null

# like 연산자

카페

- 검색하고자 하는 값을 명확히 모르는 경우에 사용한다.
- 유사 패턴 방식으로 검색 가능하다.
- 반드시 Wildcard(%와 \_)를 사용해야 검색한다.
- 예시) 거주지 동네 이름을 찾고자 할 때 많이 사용된다.

기호	의미
%	0개 이상의 문자
_	반드시 1개의 문자

내용	sql 구문
첫 글자가 [선] 인 사원	select name from employees where name like '선%';
이름 중에 [연]이라는 글자가 있는 사원	select name from employees where name like '%연%';
이름의 2 번째 글자가 [리] 인 사원	select name from employees where name like '_리%';
이름 중에 [성] 이 들어 있는 사원	select name from employees where name like '%성%';

# 논리 연산자 예시

카페

내용	sql 구문
급여가 500 이면서 동시에 이름에 [진] 을 포함하는 사원	<pre>select sabun, name, salary from employees where salary = 500 and name like '%진%' ;</pre>
급여가 500 이거나 이름에 [진] 을 포함하는 사원	<pre>select sabun, name, salary from employees where salary = 500 or name like '%진%' ;</pre>
managerid가 2이면서 이름이 [대성], [승리] 를 포함하지 않는 사원	<pre>select name, managerid from employees where managerid = 2 and name not in('대성', '승리') ;</pre>
이름에 [리]라는 글자를 포함하거나 급여가 400 이상인 사원	<pre>select name, salary from employees select name, salary from employees where name like '%리%' or salary &gt;= 400 ;</pre>
급여가 100 이하이고 , 이름이 [박] 과 [이] 사이에 존재하는 사원	<pre>select name, salary from employees where (name between '박' and '이' ) and salary &lt;= 200 ;</pre>



# 비교 연산자 예시

카페

- 원활한 실습을 하기 위하여 다음 문장을 수행한다.
  - 사번을 5로 나눈 값에 100 곱하기
  - `update employees set salary = 100 * (mod(sabun, 5) + 1) ;`
- `commit ;`

내용	sql 구문
급여가 200 이하인 사원 조회	<code>select name, salary from employees where salary &lt;= 200 ;</code>
200 과 300 사이의 급여를 받는 사원	<code>select name, salary from employees where salary between 200 and 300 ;</code>
이름이 [ 이 ] 과 [ 지 ] 사이의 값	<code>select name from employees where name between '이' and '지' ;</code>
사번이 1, 10, 15 인 사원들	<code>select sabun, name, salary from employees where sabun in(1, 10, 15) ;</code>
이름이 ' 제시카', ' 티파니' 인 사원	<code>select sabun, name from employees where name in('제시카', '티파니') ;</code>
모든 관리자를 조회하시오 즉, managerid 가 null 인 사원	<code>select name from employees where managerid is null ;</code>

# ESCAPE 연산자

- like 연산자 사용시 %나 \_에 대해서는 예외 처리된다
- %나\_를 데이터로 인식하기 위해서는 ESCAPE 키워드 사용
- 사용 형식
  - where 컬럼 LI KE 'X%X\\_X'ESCAPE '\'

# 연산자 우선 순위

카페

- 괄호를 사용하게 되면 연산자의 우선 순위를 임의로 변경할 수 있다.

Operator	Meaning
1	산술 연산자(Arithmetic operators)
2	결합 연산자(Concatenation operator)
3	비교 연산자(Comparison conditions)
4	IS [not] null, like, [not] IN
5	[not] between
6	not equal to
7	not logical condition
8	AND logical condition
9	OR logical condition

# order by

카페

정렬	ASC(오름차순)	DESC(내림차순)
숫자	1 2 3	3 2 1
문자	A B C	C B A
날짜	2009-01-03 2009-02-22 2009-04-15	2009-04-15 2009-02-22 2009-01-03
NULL	가장 마지막에 나온다.	가장 먼저 나온다.

# order by 예시

- 원활한 실습을 하기 위하여 다음 문장을 수행한다
  - update employees set salary = null where sabun in(4, 6, 12) ;
  - commit ;

내용	sql 구문
사번과 이름과 급여를 조회하되 사번 순으로 정렬하시오.	select sabun, name, salary from employees order by sabun [asc] ;
사번과 이름과 급여를 조회하되 사번의 역순으로 정렬하시오.	select sabun, name, salary from employees order by sabun desc ;
높은 급여를 받는 순서대로 사원을 조회하되 , 동일한 급여에는 사번이 높은 순으로 정렬하기	select sabun, name, salary from employees order by salary desc, sabun desc ;
사원의 이름과 급여와 연봉을 조회하되 연봉의 역순으로 정렬하기 Alias 를 이용한 정렬하기	select name, salary, 12 * salary as annsal from employees order by annsal desc ;
첫 번째 컬럼으로 오름차순 정렬하기	select name, salary from employees order by 1 ;

# 치환 변수

카페 1059

- 비슷한 형태의 반복적인 쿼리 문장의 변수를 외부에서 입력 받아서 처리하고자 할 때 사용한다.
- 사용 예시
  - `select * from employees where sabun = 10 ;`
  - `select * from employees where sabun = 20 ;`
- 다음과 같은 구문에 사용이 가능하다.
  - where 절
  - order by 절
  - 컬럼 표현식
  - 테이블 이름
  - select의 모든 구문

변수	의미
&	일회성 변수로써 이번 문장에서만 사용 가능하다.
&&	해당 세션이 유지되는 동안 값을 저장하기 위한 변수를 의미한다. 세션이란 사용자가 DB에 접속한 시점부터 종료하기 까지의 기간을 말한다.

# 치환 변수 예시

카페 1059

& 사용 예시	&& 사용 예시
<pre>select sabun, name, salary, hiredate from employees where sabun = &amp;empno ;</pre> <p>1, 5, 10 등등 입력해보기</p>	<pre>select sabun, name, salary, hiredate from employees where sabun = &amp;&amp;empno ;</pre> <p>1, 5, 10 등등 입력해보기</p>
<pre>select sabun, name, salary, salary*12 from from employees where name = '&amp;ename' ;</pre> <p>[제시카] 입력해보기</p>	<pre>select sabun, name, salary, salary*12 from from employees where name = '&amp;&amp;ename' ;</pre> <p>[제시카] 입력해보기</p>
<pre>select sabun, name, salary, &amp;colname from employees where &amp;condition order by &amp;order_condition ;</pre> <p>각각 hiredate, salary &gt;= 500, sabun 입력해 보기</p>	<pre>select sabun, name, salary, &amp;&amp;colname from employees where &amp;&amp;condition order by &amp;&amp;order_condition ;</pre> <p>각각 hiredate, salary &gt;= 500, sabun 입력해 보기</p>

# DEFINE/VERIFY

카페 2344

- DEFINE : 변수의 값을 외부에서 미리 선언하고자 하는 경우 사용
- VERIFY : 치환 변수 전후의 상태 값 보기/숨기기

형식	의미
DEFINE	변수 목록 보기
DEFINE 변수 A	변수 A 정보 보기
DEFINE 변수 A = 값	변수 A 에 값 정의 하기
UNDEFINE 변수 A	변수 A 해제

형식	의미
SET VERIFY ON	치환 변수 전후 상태 보기
SET VERIFY OFF	치환 변수 전후 상태 보이지 않기

## 사용 예시

-- 변수 설정

```
define empnum=5
```

-- verify 옵션 기능 끄기

```
set verify off
```

```
select sabun,name, salary  
from employees  
where sabun = &empnum ;
```

-- 변수의 값 해제

```
undefine empnum
```

-- verify 옵션 기능 켜기

```
set verify on
```



# Accept 사용법

- 사용 방법

- ACCEPT variable [datatype][FORMAT format][PROMPT string]{hide}

항목	설명
datatype	char, number, date 데이터 타입중 하나를 선택, 기본값은 char
format	문자 상수 형태이며 데이터 타입이 date 일 경우 'yyyy/mm/dd' 형태도 가능 A10, 9,999 등등의 표시 형식 지정이 가능하다
prompt	변수 값을 입력 받기 전에 문자열을 보여줌
hide	프롬프트 상태에서 값을 입력시 입력하는 값이 화면에 보이지 않도록 함

# Oracle(SQL)

함수(function)



# DUAL 테이블

카페

항목	설명
정의	SYS 관리자가 소유하고 있는 1행 1열의 테이블
특징	모든 USER가 사용할 수 있다.
용도	계산기 : 간단한 계산식 필요시 사용 ( 실 테이블 필요 없음 ) 함수 검증시에도 사용된다.
사용 예시	<pre>DESC DUAL 이름 널? 유형 ----- DUMMY VARCHAR2(1)  select * from DUAL ; select 2 * 3 from DUAL; select 5 * 5 * 5 from DUAL;  -- power(x, y) : x의 y승 -- power(2, 10) : 2의 10승 = 1024 select POWER(2, 10) from DUAL ; -- 2의 10승 = 1024  select mod(12, 5) from dual ;</pre>

# 단일행 함수의 특징

카페

- 데이터 값을 조작하는데 사용된다.
- 하나 이상의 인수(argument)들을 받고 하나의 결과를 리턴한다.
- 데이터 수정과 중첩이 가능하다.

함수의 종류	함수 예시
문자 함수	LOWER, UPPER, INITCAP, CONCAT, SUBSTR, LENGTH, INSTR, LPAD/RPAD, REPLACE, TRIM/LTRIM/RTRIM
숫자 함수	ROUND, TRUNC, MOD
날짜 함수	MONTHS_between, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
변환 함수	TO_CHAR, TO_NUMBER, TO_DATE
일반 함수	NVL, NVL2, nullIF, COALESCE, CASE, DECODE

# 문자열 함수

카페

함수의 종류	의미
LOWER	소문자로 변경 , LOWER( 컬럼  표현식 )
UPPER	대문자로 변경 , UPPER( 컬럼  표현식 )
INITCAP	각 단어의 첫 글자만 대문자 변경 , INITCAP( 컬럼  표현식 )
CONCAT	두 개의 문자 연결, CONCAT( 컬럼 1 표현식 1, 컬럼 2 표현식 2)
SUBSTR	문자열의 부분 집합 추출 , SUBSTR( 컬럼  표현식, m, [n])
LENGTH	문자열의 길이, LENGTH( 컬럼  표현식 )
INSTR	지정한 문자의 위치를 숫자 값으로 리턴, INSTR( 컬럼  표현식 , 찾을문자 [, 시작위치 ])
LPAD	오른쪽 정렬 후 왼쪽에 생긴 빈 공백에 특정 문자를 채운다. 지정 글자 채워 넣기, LPAD( 컬럼  표현식 , 채울숫자 , 채울문자 )
RPAD	왼쪽 정렬 후 오른쪽에 생긴 빈 공백에 특정 문자를 채운다.
REPLACE	문자열 치환 함수, REPLACE( 대상문자열 , 치환될문자열 , 치환할문자열 )
LTRIM	왼쪽에서 특정 문자를 삭제한다.
RTRIM	오른쪽에서 특정 문자를 삭제한다.

문자\_함수\_실습.txt

# 숫자 함수

카페

함수의 종류	의미
ROUND	숫자 값을 반올림, ROUND(컬럼   표현식, n)
TRUNC	숫자 값을 절삭, TRUNC(컬럼   표현식, n)
MOD	나머지 반환 함수, MOD(m n)

숫자\_함수\_실습.txt

# 날짜 함수

카페

- 오라클 내장 함수 sysdate는 [현재 시간]을 알려 주는 함수이다.
- NLS\_DATE\_FORMAT
  - 세션 관련 날짜 형식을 변경하기 위하여 사용하는 환경 변수이다.
  - 현재 시간을 어떠한 서식으로 보여 줄인것인가? 를 나타내는 환경 변수
  - 날짜 데이터 포맷을 변경하는 방법
    - ALTER SESSION SET NLS\_DATE\_FORMAT = 'YYYY-MM-DD HH:MI:SS';
- 1日 = 24HR = 1440分 = 86400초
- 5분 →  $5 * 1 / 24 / 60$

연산의 종류	결과	설명
날짜+숫자	날짜	일수에 날짜 더하기
날짜-숫자	날짜	날짜에서 일수 빼기
날짜-날짜	숫자	한 날짜에서 다른 날짜 빼기
날짜+숫자/24	날짜	날짜에 시간 더하기

함수의 종류	의미
months_between	두 날짜 사이의 월수를 반환
add_months	날짜에 월 더하기
next_day	명시한 날짜 이후의 요일의 날짜
last_day	해당 월의 마지막 날
round	[형식 모델]을 기준으로 반올림
trunc	[형식 모델]을 기준으로 절삭(무조건 버림)

날짜\_함수\_실습.txt

# 변환 함수

카페

함수의 종류	의미
TO_CHAR	[날짜형] 혹은 [숫자형]을 [문자형] 데이터로 변환한다.
TO_DATE	[문자형]을 [날짜형] 데이터로 변환한다.
TO_NUMBER	[문자형]을 [숫자형] 데이터로 변환한다.

변환\_함수\_실습.txt



# TO\_CHAR(문자형)

카페

- TO\_CHAR( 날짜 데이터, 형식\_요소 )

형식 요소 종류	의미	사용 예시
YYYY	년도 표현(4자리)	2014
YY	년도 표현(2자리)	14
MM	월을 숫자로 표현	03
MON	월을 알파벳으로 표현	JAN
MONTH	월의 완전한 이름	JANUARY
DAY	요일 표현	SUNDAY
DY	요일을 약어로 표현	SUN
D	주의 일수	5
AM 또는 PM	오전(AM), 오후(PM) 시각 표시	-
HH 또는 HH2	시간(1~12)	-
HH24	24시간으로 표현(0~23)	-
MI	분 표현	-
SS	초 표현	-
RN	로마식 달(月) 수 표현	XI

# TO\_CHAR(문자형)

카페

형식 요소 종류	의미	사용 예시	결과
9	한 자리의 숫자 표현	(1111, '99999')	1111
0	앞부분을 0으로 표현	(1111, '099999')	001111
\$	달러 기호를 앞에 표현	(1111, '\$99999')	\$1111
.	소수점을 표시	(1111, '99999.99')	1111.00
,	특정 위치에 , 표시	(1111, '99,999')	1,111
MI	오른쪽에 -기호 표시	(1111, '99999MI')	1111-
PR	음수 값을 <>으로 표현	(1111, '99999PR')	<1111>
EEEE	과학적으로 표기법으로 표현	(1111, '9.999EEEE')	1.111E+03
V	10n을 곱한 값으로 표현	(1111, '999V99')	111100
B	공백을 0으로 표현	(1111, 'B9999.99')	1111.00
L	지역 통화(Local currency)	(1111, 'L99999')	\1111

# 일반 함수

카페

함수의 종류	의미
NVL	NVL(식1, 식2) : [식1]이 널이면 [식2]로 대체
NVL2	NVL2 (널여부, 널이아닐때수행됨, 널일때수행됨) 프로그래밍의 if ... else 구문과 동일하게 동작한다.
nullIF	nullIF (식1, 식2) : [식1]과 [식2]가 같으면 널을, 아니면 [식1]을 취함
COALESCE	COALESCE (식1, 식2, , 식n) 리스트 중에서 첫 번째로 널이 아닌 값을 취함 열거하는 [식]들의 타입이 일치해야 한다 프로그래밍의 다중 if 구문과 동일하게 동작함

## • 실습 사전 준비

- update employees set salary = 100 \* (mod(sabun, 5) + 1) ;
- update employees set salary = null where sabun in(1, 2, 3) ;
- update employees set email = 'abc@xx.com' ;
- update employees set email = null where sabun in(3, 4, 5) ;
- -- 사번 3은 메일과 급여 모두 null이다.
- commit ;

# 일반 함수

내용	sql 구문
모든 사원의 이름과 급여를 조회하시오. 단, 급여가 없는 사원은 기본 값이 50원이다.	select name, nvl(salary, 50) from employees order by sabun asc ;
모든 사원의 이름과 급여와 연봉을 조회하시오. 단, 급여가 없는 사원은 기본 값이 50원이다. 연봉 : $12 * \text{급여} = 12 * 50 = 600$	select name, nvl(salary, 50), 12 * nvl(salary, 50) as 연봉 from employees order by sabun asc ;
모든 사원의 이름과 급여와 급여 존재 여부를 조회 하시오. 단, 급여 존재 여부는 nvl2 함수를 사용해 보세요.	select name, salary, nvl2(salary, '급여 존재함', '급여가 없음') as 결과 from employees order by sabun ;
모든 사원의 이름과 급여와 상태를 조회하시오. 단, 상태는 급여가 300인 직원들은 null 값으로 처리 하시오.	select name, salary, nullif(salary, 300) as 결과 from employees ;
-	col 결과 for a15 select name, salary, email, coalesce(to_char(salary), email, '둘다 없음') as 결과 from employees order by sabun ;
모든 사원의 이름과 이메일을 조회 단, 메일 주소가 없으면 [메일 없음]이라고 표현하시오.	select name, nvl(email, '메일 없음') as email from employees order by sabun ;
모든 사원의 이름과 보너스를 조회 단, 보너스는 급여의 10%이고 급여가 없으면 99원이라고 가정한다.	select name, salary, nvl2(email, 0.1*salary, 99) as bonus from employees order by sabun ;

일반\_함수\_실습.txt

# 함수의 중첩

카페

- 단일 행 함수는 여러 레벨에 걸쳐서 중첩이 가능하다.
- 계산 순서 : 가장 하위 레벨에서 상위 레벨 순으로 계산된다.
- 사용 형식
  - 함수 3 ( 함수 2 ( 함수 1(COL, 인수 1), 인수 2), 인수 3)

## 사용 예시

```
select last_name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
from employees  
where department_id =60;
```

# Case 표현식

카페 1060

- SQL 문장에서 IF-THEN-ELSE와 같은 흐름 제어문 역할을 수행한다.

내용	sql 구문
사용 형식	<pre>case 조건식   when 비교_표현식1 then 리턴표현식1   [ when 비교_표현식2 then 리턴표현식2     when 비교_표현식n then 리턴표현식n     else else_표현식   ]</pre>
사용 예시	<p>매니저 아이디가 1 이면 '소녀시대', 2 이면 '빅뱅', 3 이면 '원더걸스' 이외에는 '관리자'라고 표현하는 case 표현식을 작성하시오.</p> <pre>select name, managerid,        case managerid          when 1 then '소녀시대'          when 2 then '빅뱅'          when 3 then '원더걸스'          else '관리자' end as result from employees ;</pre>

# DECODE 함수

카페 1060

- case나 if-then-else 문장의 조건적 조회를 가능하게 해준다.

내용	sql 구문
사용 형식	<pre>decode( 표현식 ,         조건 1, 값 1,         조건 2, 값 2,         조건 3, 값 3,         ...         기타일경우의값       )</pre>
사용 예시	<pre>매니저 아이디가 1 이면'소녀시대', 2 이면'빅뱅', 3 이면'원더걸스' 이외에는'관리자'라고 표현하는 case 표현식을 작성하시오 select name, managerid,        decode(managerid,               1, '소녀시대',               2, '빅뱅',               3, '원더걸스',               '관리자') as result from employees ;</pre>

# Oracle(SQL)

그룹 함수





# 그룹(집계) 함수

카페

- 그룹 함수는 그룹 당 하나의 결과를 리턴한다
- 원활한 실습을 위하여 다음 문장을 수행하도록 한다
  - `update employees set salary = 100 * sabun ;`
  - `commit ;`
  - `-- 999999는 6자리의 숫자를 의미함`
  - `col sabun for 999999`
  - `col name for a12`
  - `col salary for 999999`
  - `select sabun, name, salary, hiredate from employees ;`

그룹 함수	설명	숫자	문자/날짜
avg	평균, 널 값 무시	0	X
count	count(*) : null 값 포함하여 전체 행수 표현 count(expr) : null 값 제외한 개수 표현 count(distinct expr) : 중복된 값을 제외한 것 중 null 값을 갖지 않는 것	-	-
max	최대 값, 널 값 무시	0	0
min	최소 값, 널 값 무시	0	0
sum	합계, 널 값 무시	0	X

# 그룹 함수 사용 예시

카페

내용	sql 구문
모든 사원들의 급여의 총합을 구하시오.	select <b>sum</b> (salary) as sumsal from employees ;
모든 사원들의 급여의 평균을 구하시오.	select <b>avg</b> (salary) as avgsal from employees ;
최대 급여자는 누구인가 ?	select <b>max</b> (salary) from employees ;
최소 급여자는 누구인가 ?	select <b>min</b> (salary) from employees ;
이름 순으로 정렬시 가장 먼저 나오는 사람은 누구인가 ?	select <b>min</b> (name) from employees ;
이름 순으로 정렬시 가장 나중에 나오는 사람은 누구인가 ?	select <b>max</b> (name) from employees ;
다음 문장을 수행하라.	update employees set salary = null where sabun in(4, 6, 10) ; commit ;
모든 사원의 수는 ? (총 행수)	select <b>count</b> (*) from employees ;
급여가 null이 아닌 사원의 수는 ?	select <b>count</b> (salary) from employees ;

# group by 기본 문형

카페

## select 기본 문법

```
select [column,] group_function(column),  
from table  
[where condition]  
[group by column]  
[HAVING Group_Condition]  
[order by column];
```

## 그룹핑 사용 지침

- 1) 그룹화할 컬럼을 결정하여 group by 절에 명시하라
  - 2) 동일한 컬럼들을 select 절에 명시하라
  - 3) 요구한 그룹 함수를 select 절에 명시하라
- ◆ select 절의 그룹 함수가 아닌 컬럼은 무조건 group by 에 포함
  - ◆ group by 절에 Alias 사용 불가 ( 반드시 컬럼명 )
  - ◆ where : 조건절( 일반 컬럼 , 단일행 함수 )
  - ◆ Having : 조건절(그룹 함수에 대한 조건 / 일반 함수 조건 )

# 그룹 함수 사용 예시

카페

내용	sql 구문
이름과 매니저 아이디 조회	<code>select name, managerid from employees ;</code>
매니저별 모든 직원들의 수를 구하시오.	<code>select managerid , count(name) from employees group by managerid ;</code>
매니저별 모든 직원들의 수를 구하되, managerid가 존재하지 않는 직원들은 배제하시오.	<code>select managerid , count(name) from employees where managerid is not null group by managerid ;</code>
매니저별 직원들의 급여의 총합을 구하시오.	<code>select managerid , sum(salary) from employees group by managerid ;</code>
매니저별 직원들의 급여의 평균을 구하시오.	<code>select managerid , avg(salary) from employees group by managerid ;</code>
각 매니저별 모든 직원들을 이름을 출력할 때 가장 먼저 /나중에 나오는 사람은 각각 누구인가 ?	<code>select managerid , min(name) min, max(name) max from employees group by managerid ;</code>
매니저별 직원들의 급여의 총합을 구하되, 총 금액이 6000 이상인 항목만 조회하시오.	<code>select managerid , sum(salary) from employees group by managerid <b>having</b> sum(salary) &gt;= 6000 ;</code>
매니저별 직원들의 급여의 총합과 평균을 구하되, 평균 금액이 1500 이상인 항목만 조회하시오.	<code>select managerid , sum(salary), avg(salary) from employees group by managerid <b>having</b> avg(salary) &gt;= 1500 ;</code>

# 무엇이 문제인가요 ?

내용	sql 구문
무엇이 잘못된가?	select managerid, count(name) from employees ;
분석	select 절의 일반 컬럼은 반드시 group by 절에 포함이 되어야 한다
해결 방법	<p>1) managerid 컬럼을 없앤다. 그러면, 전체 사원 수가 된다.</p> <p>2) 그룹핑을 하기 위한 것이라면 group by managerid 항목을 추가해 준다. 그러면, 해당 매니저별 관리하는 직원들의 수가 된다.</p>

내용	sql 구문
무엇이 잘못된가?	select managerid , sum(salary) from employees where sum(salary) >= 6000 group by managerid ;
분석	그룹 함수는 where 절에 사용 불가능하다. 대신에 having 절을 사용하라.
해결 방법	<p>1) having 절을 사용</p> <p>select managerid , sum(salary) from employees group by managerid having sum(salary) &gt;= 6000 ;</p>

# Oracle(SQL)

조인(Join)



# 조인(Join)

카페

- alter table employees add (deptno number(6)) ;
- update employees set deptno = 10 where managerid = 1 ;
- update employees set deptno = 20 where managerid = 2 ;
- update employees set deptno = 30 where managerid = 3 ;
- update employees set deptno = 10 where name = '이수만' ;
- update employees set deptno = 20 where name = '박진영' ;
- update employees set deptno = 30 where name = '양현석' ;
- 
- --부서 테이블
- create table depts(
  - deptno number(6) primary key,
  - dname varchar2(30),
  - sales number) ;
- 
- insert into depts values(10,'sm','30000') ;
- insert into depts values(20,'jyp','50000') ;
- insert into depts values(30,'yg','70000') ;
- commit ;

조인 실습을 위하여 위의 문장들을 수행한다.

# 조인(Join)

카페

- -- 등급 테이블
- create table grades(
  - glevel varchar2(4),
  - lowsal number,
  - highsalsal number
- );
- insert into grades values('A', 0, 499) ;
- insert into grades values('B', 500, 999) ;
- insert into grades values('C', 1000, 1499) ;
- insert into grades values('D', 1500, 1999) ;
- insert into grades values('E', 2000, 10000) ;
- commit ;
- col dname for a12
- select \* from depts ;
- select \* from grades ;

col dname for a12

select \* from depts ;

DEPTNO	DNAME	SALES
-----		
10	sm	30000
20	jyp	50000
30	yg	70000

select \* from grades ;

GLEVEL	LOWSAL	HIGHSAL
-----		
A	0	499
B	500	999
C	1000	1499
D	1500	1999
E	2000	10000

조인 실습을 위하여 위의 문장들을 수행한다.



# 참조 무결성 제약 조건

카페

용어	설명
master 테이블	main( 부모 ) 테이블 Primary key : 테이블의 행을 구분하기 위한 유일한 (unique) 식별자 컬럼
detail 테이블	sub( 자식 ) 테이블 FOREIGN KEY : Master 테이블의 Primary key 를 참조하고 있는 컬럼

## 참조 무결성 제약 조건

- 부모 테이블은 자식 테이블에서 사용중인 데이터는 삭제가 불가능하다.
- 자식 테이블은 부모 테이블이 가지고 있는 값 또는 null 값을 가질 수 있다.

사원 테이블 : CHILD KEY( 자식 키 값 )

사번	이름	부서 번호
4	제시카	30
11	써니	50
99	김철수	null

부서 테이블 : PARENT KEY( 부모 키 값 )

부서 번호	소속사	총 매출액
10	sm	30,000
20	jyp	20,000
30	yg	40,000
40	abcd	1,000

50 부서는 존재하지 않는다.

# 조인의 개념

카페

```
select name, salary, deptno from employees;
```

	NAME	SALARY	DEPTNO
1	지용	1400	20
2	대성	1500	20
3	홍길동	100	(null)
4	박진영	200	20
5	양현석	300	30
6	제시카	400	10
7	티파니	500	10
8	수영	600	10
9	효연	700	10
10	서현	800	10
11	유리	900	10
12	유리	1000	10
13	써니	1100	10
14	윤아	1200	10
15	탐	1300	20
16	승리	1600	20
17	태양	1700	20
18	선미	1800	30

```
select * from depts;
```

	DEPTNO	DNAME	SALES
1	10	sm	30000
2	20	jyp	50000
3	30	yg	70000

[대성]의 소속사 이름은 ?  
그룹 jyp 의 멤버수는 몇 명인가 ?

문제 1 : [대성]의 소속사 이름은 ?  
문제 2 : 그룹 jyp 의 멤버수는 몇 명인가 ?

-- 오라클 조인 : 콤마(,) 사용      where 절 사용  
-- Ansi 조인 :    join 사용          on 절 사용

-- Ansi : 데이터 베이스 브랜드 들의 표준을 잡기 위한 협회

-- 조인 문장  
-- 모든 사원의 이름과 부서 번호, 부서명을 조회

```
select employees.name, depts.deptno, depts.dname
from employees, depts
where employees.deptno = depts.deptno
order by depts.deptno ;
```

# 조인(Join)

카페

- 둘 이상의 테이블을 쿼리하여 결과 집합 생성하는 것을 말한다.
- 한 개 이상의 테이블을 연결하여 데이터를 조회하는 것이다.
- 여러 테이블에서 특정 열을 선택한다.
- 테이블 간의 조인은 **primary key**와 **foreign key**를 이용하여 조인한다.
- where 절 내부에 두 테이블의 공통되는 컬럼을 비교한다.
- 하나 이상의 테이블에 똑같은 컬럼 이름이 있을 때 [테이블이름.컬럼이름]의 형태로 구분을 한다.

종류	종류	설명
연산자 방식	Equi Join	동일한 의미의 컬럼을 기준으로 = 연산으로 조인하는 것을 말한다.
	Non-Equi Join	동일 컬럼이 없이 non-Equal 연산자로 조인하는 것을 말한다.
내부 처리 방식	Inner Join	조인 조건을 만족하는 행만 표시한다.
	Outer Join	조인 조건에 만족하지 않는 행도 나타낸다.
Self Join	Self Join	동일 테이블끼리의 조인 방식을 말한다. Inner이면서, Equi Join 방식이다.
Cross Join	Cross Join	두 테이블의 곱(진정한 의미의 조인은 아님)

# Cross Join

- 조인시 조건 절을 지정하지 않는 조인.
- where 절이 없는 경우의 조인.
- 아무런 의미 없이 테이블을 조합해 놓은 것.
- 개발자의 실수로 발생한 경우가 대부분이다.

Oracle 조인	ANSI 조인
<pre>select * from employees, depts ;</pre>	<pre>select from employees cross join depts ;</pre>

# Equi Join

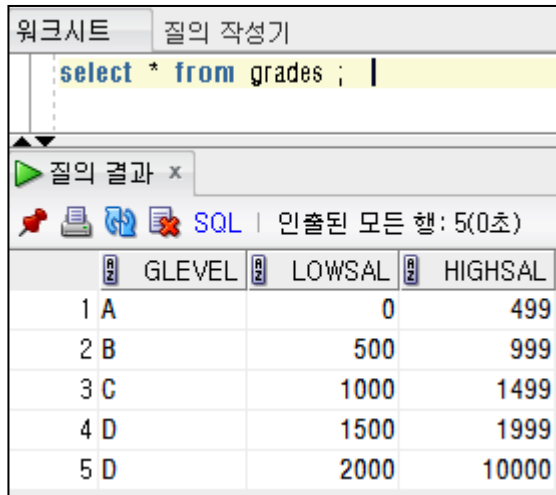
카페

- 가장 많이 사용하는 조인으로써 두 테이블에서 공통적으로 존재하는 컬럼을 연결시킨다
- 이때 비교 연산자 [=]을 사용한다.
- 컬럼 이름이 동일하면 반드시 테이블 이름을 명시하도록 한다.
- 테이블의 Alias(AS 사용 불가) 는 from 절에 명시할 수 있는데 일단 별칭이 선언되고 나면 반드시 / 무조건 별칭을 사용해야 한다.
- 추가적인 조건은 and 연산자를 사용한다.

항목	Oracle 조인	Ansi 조인
= 연산자를 이용한 조인	<pre>select employees.name, depts.dname from employees, depts where employees.deptno = depts.deptno ;</pre>	<pre>select employees.name, depts.dname from employees inner join depts on employees.deptno = depts.deptno ;</pre>
Alias를 사용	<pre>select e.name, d.dname from employees e, depts d where e.deptno = d.deptno ;</pre>	<pre>select e.name, d.dname from employees e inner join depts d on e.deptno = d.deptno ;</pre>
추가적인 조건이 들어 가는 경우 and 연산자를 이용하라	<pre>select e.name, d.dname from employees e, depts d where e.deptno = d.deptno and e.name in('제시카', '티파니') ;</pre>	<pre>select e.name, d.dname from employees e inner join depts d on e.deptno = d.deptno and e.name in('제시카', '티파니') ;</pre>

# Non-Equi Join

- 테이블의 컬럼 값이 직접적으로 일치하지 않는 경우에 사용하는 조인이다.
- 이름에서 유추 가능 하듯이 [=]연산자가 아닌 다른 종류의 연산자를 사용한다.
- 통상적으로 between 또는 in 절이 많이 사용 된다.



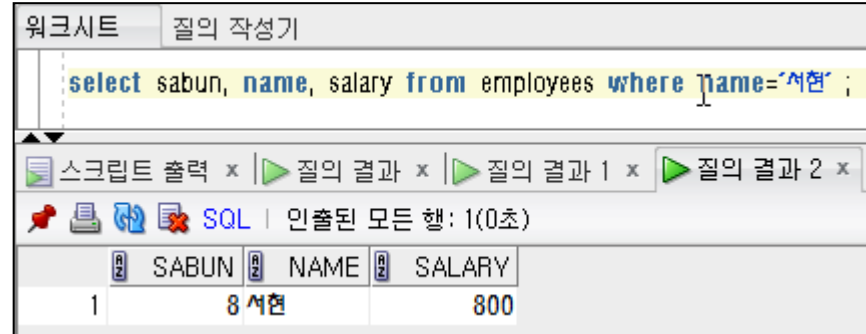
워크시트 | 질의 작성기

```
select * from grades ;
```

질의 결과 x

SQL | 인출된 모든 행: 5(0초)

	GLEVEL	LOWSAL	HIGHSAL
1 A		0	499
2 B		500	999
3 C		1000	1499
4 D		1500	1999
5 D		2000	10000



워크시트 | 질의 작성기

```
select sabun, name, salary from employees where name='서현' ;
```

스크립트 출력 x | 질의 결과 x | 질의 결과 1 x | 질의 결과 2 x

SQL | 인출된 모든 행: 1(0초)

	SABUN	NAME	SALARY
1	8	서현	800

[서현]이는 하한 값 500과 상한 값 999 사이에 있으므로 레벨이 [B] 이다.


```
update employees set salary = 800 where name='서현' ;  
commit ;
```

```
select e.name, e.salary, g.glevel  
from employees e, grades g  
where e.salary between g.lowsal and g.highsal  
and e.name = '서현' ;
```

# Outer Join

카페

- 단순히 join이라 함은 inner join을 의미한다.
- Inner Join + alpha(단,  $\alpha \geq 0$ )
- 조인 조건에 만족하지 않는 행들도 나타내기 위함이다.
- Oracle 조인은 (+)을 사용한다(Full Outer Join은 제외)
- outer join은 반드시 방향과 outer 키워드를 명시해야 한다.

항목	Oracle 조인	Ansi 조인
right outer 조인	<pre>select e.name, d.dname from employees e, depts d where e.deptno(+) = d.deptno ;</pre>	<pre>select e.name, d.dname from employees e right outer join depts d on e.deptno = d.deptno ;</pre>
left outer 조인	<pre>select e.name, d.dname from employees e, depts d where e.deptno = d.deptno(+);</pre>	<pre>select e.name, d.dname from employees e left outer join depts d on e.deptno = d.deptno ;</pre>
full outer 조인	- 	<pre>select e.name, d.dname from employees e full outer join depts d on e.deptno = d.deptno ;</pre>

# Self Join

- Equi Join을 이용하여 자기 자신의 테이블과 조인하는 것을 의미한다.
- 반드시 테이블은 Alias를 사용해야만 구분이 된다.
- 예시
- 사원의 이름과 그의 매니저 이름을 한 줄에 출력하시오
- 예시 : [대성]의 매니저는 [박진영]이다
- 사원\_테이블.관리자\_아이디 = 관리자\_테이블.사번
- -- self join
- select employees.name, manager.name
- from employees inner join employees manager
- on employees.managerid = manager.sabun ;

워크시트 | 질의 작성기

```
select sabun, name, managerid from employees
where managerid is null or managerid = 2 ;
```

질의... x

SQL | 인출된 모든 행: 8(0초)

	SABUN	관리자 테이블
1	14	
2	15 대성	2
3	1 홍길동	(null)
4	2 박진영	(null)
5	3 양현석	(null)
6	13 탐	2
7	16 승리	2
8	17 태양	2

질의...

	SABUN	NAME	MANAGERID
1	14	지용	2
2	15 대성		2
3	1 홍길동		(null)
4	2 박진영		(null)
5	3 양현석		(null)
6	13 탐		2
7	16 승리		2
8	17 태양		2



# Natural Join

- Natural 키워드를 사용하여 조인하는 방법을 말한다.
- 2 개의 테이블에 동일 이름/데이터 타입의 컬럼이 반드시 1개 있어야만 가능하다.
- 공통 컬럼을 조사하여 내부적으로 조인문을 생성한다.
- 양쪽 테이블에 동일한 이름과 동일한 타입의 컬럼이 존재하는 경우 **자연스럽게 조인**하겠다.
- natural join
  - select employees.name, depts.dname
  - from employees natural join depts;
  - -- managerid가 1인 사원만 natural join
  - select e.name, d.dname
  - from employees e natural join depts d
  - where managerid = 1 ;

# Using 절을 이용한 조인

- ON 절 대신에 Using 절을 사용할 수 있다.
- Using 절은 절대로 별칭이 불가능하다.
- Oracle 조인은 지원하지 않는다.
- 동등 조인을 간편하고 쉽게 표현할 수 있다
- ON 절보다는 더 이해하기 쉽다.
- 사용 예시
  - select e.name, d.dname
  - from employees e inner join depts d
  - using(deptno);

# Oracle(SQL)

서브 쿼리(subQuery)



# 서브 쿼리

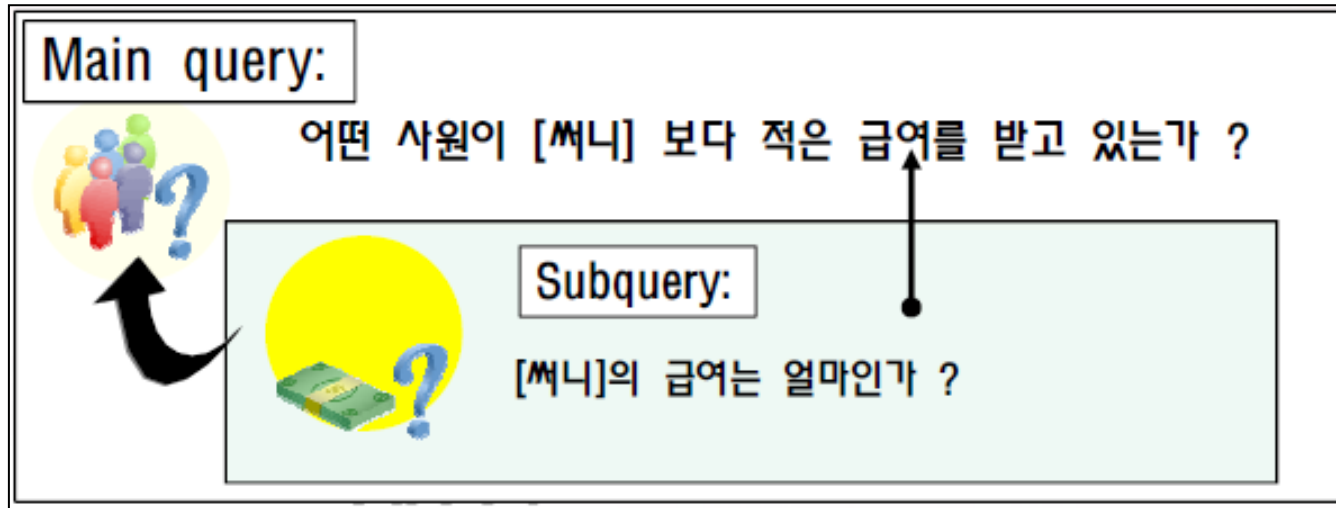
카페 338

- [씨니] 보다 적은 급여를 받고 있는 사원은 누구인가요 ?
- 작업 순서 ( 씨니는 사번이 11번이라고 가정한다. )
  - 1) 씨니의 급여를 구한다
    - `select salary from employees where sabun =11 ;`
    - 결과 : 1100 원
  - 2) where 절에 급여를 대입해본다
    - `select name, salary from employees where salary < 1100 ;`
  - 3) 1)과 2)의 내용을 합친다
    - `select name, salary from employees`  
`where salary <`  
`(`  
`select salary from employees where sabun =11`  
`) ;`

# 서브 쿼리

카페 338

- [써니] 보다 적은 급여를 받고 있는 사원은 누구인가요 ?



[써니]의 급여는 얼마인가 ?

```
select name, salary from employees
where salary < (
    select salary
    from employees
    where sabun = 11
) ;
```

서브 쿼리의 특징

서브 쿼리는 한번만 수행이 되고, 메인 쿼리 보다 먼저 수행 반드시 괄호로 둘러 싸야 한다.  
서브 쿼리와 메인 쿼리는 동일 테이블이어도 무방하다.  
비교 연산자의 오른쪽에 위치해야 한다.  
유형 : 단일행 / 다중행 서브 쿼리  
order by 절을 사용할 수 없다 ( 단, top-n 절을 제외)

# 서브 쿼리 사용 가능 영역

카페 338

- 서브 쿼리의 종류

종류	설명
단일행 서브 쿼리	한 개의 행을 리턴한다.( 조회 결과 건수 : 1 건 ) 관련 연산자 : =( ~ 와 같다 ), >, <, >=, <=, <>( ~ 와 같지 않다 )
다중행 서브 쿼리	하나 이상의 행이 리턴된다.( 조회 결과 건수 : 2 건 이상 ) 관련 연산자 : In, Any, All

- 서브 쿼리 사용 가능 영역
- select 절의 from/where/having 절
  - DML(insert, delete, update)
  - CREATE TABLE 중에서 → 테이블 복사
  - CREATE VIEW

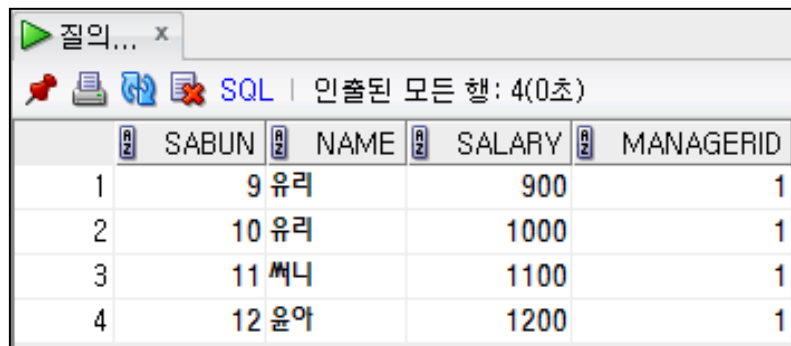
# 단일행 서브 쿼리 수행

카페 825

- 사번 7번과 동일한 managerid를 가지고 있고, 사번 8번의 사원보다 많은 급여를 받는 사원을 조회하시오.

## sql 문장

```
select sabun, name, salary, managerid  
from employees  
where managerid = ( select managerid from employees where sabun =7)  
and salary > ( select salary from employees where sabun =8) ;
```



	SABUN	NAME	SALARY	MANAGERID
1	9	유리	900	1
2	10	유리	1000	1
3	11	써니	1100	1
4	12	윤아	1200	1

# 서브 쿼리 예시

카페 825

질문 내용	sql 구문
우리 회사에서 최저 급여를 받는 사원은 누구인가요 ?	<pre>select name, salary from employees where salary = ( select min(salary) from employees ) ;</pre>
우리 직원들의 평균 급여보다 적게 받는 사람은 ? 단, 이름 역순으로 정렬하세요.	<pre>select name, salary from employees where salary &lt; ( select avg(salary) from employees ) order by name desc ;</pre>

- 무엇이 잘못된가 ?

sql 문장
<pre>select name, salary from employees where salary = (select min(salary) from employees group by deptno ) ; *</pre> <p>3 행에 오류 : ORA-01427: 단일 행 하위 질의에 2 개 이상의 행이 리턴 되었습니다.</p>



# Will This Statement Return Rows?

- 내부 쿼리 : 존재 하지 않는 사원이다.
- 외부 쿼리 : 내부 쿼리를 null 으로 취급한다.

## sql 문장

```
select name, salary  
from employees  
where  
sabun = ( select sabun from employees where name ='김철수' );
```

[김철수]는 존재하지 않는 사원이다.

선택된 레코드가 없습니다.

Subquery returns no values.

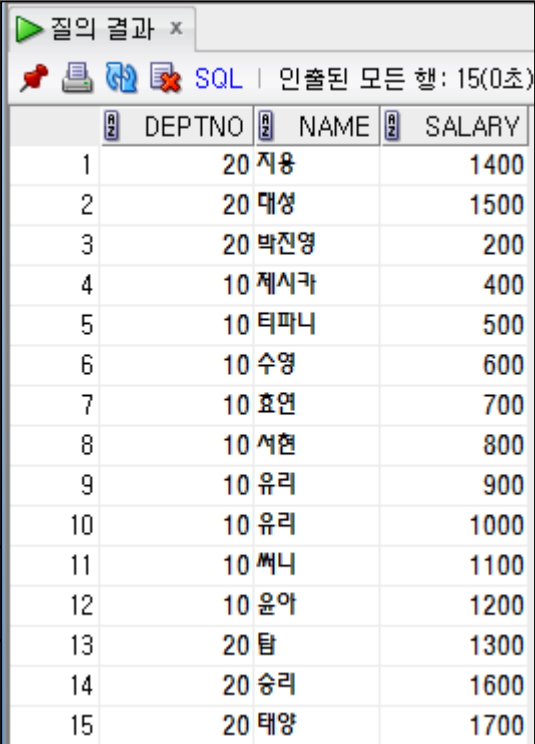
# 다중행 서브 쿼리

카페 2341

- 조회되는 결과가 2건 이상에 대한 서브 쿼리를 의미한다.
- 일반적으로 IN 연산자를 이용하여 사용한다.
- 사용 예시 :
  - 부서 이름이 [sm]이거나 [jyp]인 직원들
  - 즉, 부서 번호가 10이거나 20인 직원들을 말한다.
  - 여러 개의 행이 조회되므로 in 연산자를 이용한다.

## sql 문장

```
select deptno, name, salary
from employees
where deptno in
      ( select deptno from depts where dname in ('sm','jyp')
      ) ;
```



	DEPTNO	NAME	SALARY
1	20	지용	1400
2	20	대성	1500
3	20	박진영	200
4	10	제시카	400
5	10	티파니	500
6	10	수영	600
7	10	효연	700
8	10	석현	800
9	10	유리	900
10	10	유리	1000
11	10	써니	1100
12	10	윤아	1200
13	20	탐	1300
14	20	승리	1600
15	20	태양	1700

# Oracle(SQL)

Set(집합) 연산자



# 사전 실습 준비

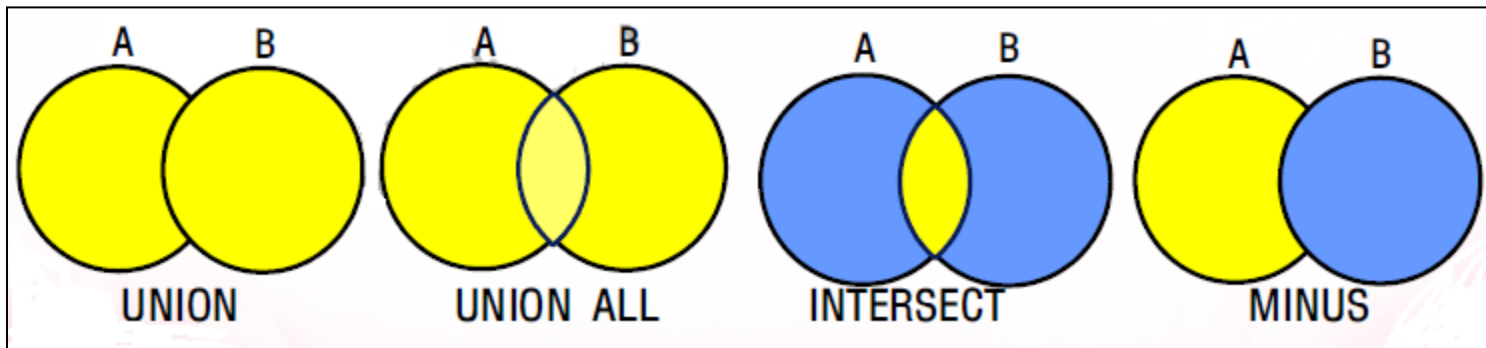
카페 339

- -- 30번 부서 번호만 저장하는 테이블 oldemp 생성
  - create table oldemp as select \* from employees where deptno = 30 ;
- -- 30번 부서 직원들을 삭제
  - delete from employees where deptno = 30 ;
- -- 사번 13번과 15번을 oldemp 테이블에 insert하기
  - insert into oldemp select \* from employees where sabun in( 13, 15 ) ;
  - commit ;
- -- 조회해보기
  - col sabun for 9999999
  - col name for a10
  - col email for a12
  - col phone for a12
  - select sabun, name, email, phone from employees ;
  - select sabun, name, email, phone from oldemp ;

# 집합(set) 연산자

카페 339

연산자	설명
UNION	각 쿼리에 의해서 선택된 결과 중에 중복 행을 제거하고 출력
UNION ALL	각 쿼리에 의해서 선택된 모든 결과를 출력, 중복 허용
INTERSECT	두 개의 쿼리를 모두 만족하는 결과 중에 중복 행을 제거하고 출력
MINUS	첫 번째 쿼리에 의해서 선택된 결과에서 두 번째 쿼리에 의해서 선택된 결과를 제거하고 출력



## • 집합 연산자 사용 지침

- 각각의 select 문장에 의해서 선택된 컬럼의 개수와 데이터 타입은 반드시 일치해야 한다.
- 집합 연산자의 처리 순서는 괄호를 이용하여 변경한다.
- order by절 사용한다면 맨 마지막에 기술하도록 한다.
- 첫 번째 쿼리 문장의 컬럼명, 별칭또는 컬럼 순번을 지정할 수 있다.
- 첫 번째 쿼리 문장의 컬럼명이 결과에 표시된다.

# UNION

카페 339

- 각각의 select 문장에 의해서 선택된 컬럼의 개수와 데이터 타입은 반드시 일치해야 한다.
- 컬럼의 이름은 같을 필요는 없다.
- UNION 연산자는 선택된 모든 컬럼들에 대하여 적용된다.
- 각각의 쿼리 결과에서 중복을 제거하는 과정에 null 값은 무시된다.
- IN 연산자가 UNION 연산자보다 우선순위가 높다.
- 기본적으로 select 문장의 첫 번째 컬럼 기준으로 오름차순 정렬이 된다

내용	sql 구문
이전 데이터 포함하여 모두 조회하시오.	<pre>select sabun, name, email, phone from employees union select sabun, name, email, phone from oldemp ;</pre>
부서 번호가 10또는 30만 조회하시오.	<pre>select sabun, name, email, phone from employees where deptno in( 10, 30 ) union select sabun, name, email, phone from oldemp where deptno in( 10, 30 );</pre>

# UNION ALL

카페 339

- UNION 과는 달리 중복된 행을 제거하지 않으며, 디폴트로 출력 결과를 정렬하지 않는다.
- distinct 키워드는 사용할 필요가 없다.
- union all 은 중복 포함이므로 사번 13 과 1 5는 각각 2건씩 중복되어 나타난다.

내용	sql 구문
이전 데이터 포함하여 모두 조회하시오 단, 데이터가 중복이 되더라도 모두 표시되도록 한다. order by 1은 '1번째 컬럼으로 정렬'하라는 뜻이다.	<pre>select sabun, name, email, phone from employees union all select sabun, name, email, phone from oldemp order by 1 ;</pre>

# INTERSECT

카페 339

- INTERSECT 연산자는 각각의 쿼리에 공통적으로 포함되어 있는 행들을 출력한다
- 각각의 select 문장에 의해서 선택된 컬럼의 개수와 데이터 타입은 반드시 일치해야 한다.
- 컬럼의 이름은 같을 필요는 없다.
- 각각의 쿼리 작성 순서는 연산 결과를 변경하지 않는다.
- INTERSECT 연산자는 null 값을 무시하지 않는다.

내용	sql 구문
양쪽 테이블에 공통으로 들어 있는 데이터를 모두 조회하시오.	<pre>select sabun, name, email, phone from employees intersect select sabun, name, email, phone from oldemp ;</pre>



# MINUS

카페 339

- MINUS 첫 번째 쿼리 결과에서 두 번째 쿼리 결과에 포함된 행들을 제거하고 출력한다.
- 각각의 select 문장에 의해서 선택된 컬럼의 개수와 데이터 타입은 반드시 일치해야 한다.
- 컬럼의 이름은 같을 필요는 없다.

내용	sql 구문
employees 테이블에 존재하는 데이터만 조회하시오.	<pre>select sabun, name, email, phone from employees minus select sabun, name, email, phone from oldemp ;</pre>
oldemp 테이블에 존재하는 데이터만 조회하시오.	<pre>select sabun, name, email, phone from oldemp minus select sabun, name, email, phone from employees;</pre>

# Oracle(SQL)

사용자 권한(DCL)



# DCL 개요

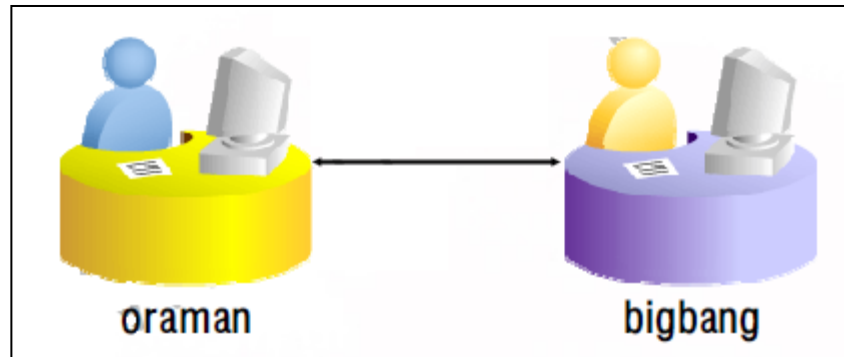
카페 452, 827

용어	설명
권한 (Privilege)	DB 내에서 작업할 수 있는 어떠한 능력 또는 범위 테이블을 삭제할 수 있다.( 삭제 권한 ) 행(row) 을 업데이트 할수 있다.( 수정 권한 )
DCL	Data Control Language 이러한 권한을 부여하고, 제어할 수 있는 명령어
관련 명령어	grant : 필요한 권한을 사용자 또는 Object 에 부여한다. revoke : 부여했던 권한을 박탈(제거)한다.
Role(롤)	1개 이상의 권한을 묶어서 그룹화 시킨 것을 말한다.

# 다른 유저의 테이블 접근

카페 452, 827

- oraman이 bigbang의 employees 테이블을 조회하려고 한다.
- bigbang이 oraman의 employees 테이블을 조회하려고 한다.
- 신입 사원 bigbang는 관리자가 생성해야 한다.



select \*  
from bigbang.employees;

select \*  
from oraman.employees;

- 접근 절차
  - 단계1 : bigbang 사용자를 생성
  - 단계2 : bigbang가 로그인 가능하도록 권한 부여
  - 단계3 : hr.employees 테이블 조회 권한을 bigbang에게 부여
  - 단계4 : bigbang 사용자는 CTAS 방법으로 employees 테이블 생성
  - 단계5 : bigbang 사용자는 employees 테이블 조회 권한을 oraman에게 부여
  - 단계6 : oraman 사용자는 employees 테이블 조회 권한을 bigbang에게 부여
  - 단계7 : 상대방 테이블 조회해보기

# 사용자 생성

카페 452, 827

- 사용자 생성은 CREATE USER 권한이 있는 자에 의해 생성이 가능하다.
- 즉, 관리자는 사용자 생성이 가능하다.
- alter database default tablespace users ; -- 기본 TBS 변경하는 문장

## 사용 형식

```
create user 아이디  
identified by 비밀번호  
quota unlimited on users  
account unlock;
```

## 사용 예시

```
create user bigbang  
identified by bigbang  
quota unlimited on users  
account unlock;
```

tablespace : table이나 index 등을 저장하기 위한 데이터 저장소

quota : tablespace에 할당 받은 공간

users 테이블 스페이스 : default 테이블스페이스 이름

account unlock : 로그인 바로 가능하게 계정을 풀어 주겠다. ↔ lock

--특정 사용자가 사용하고 있는 Default Tablespace 확인하기 ( 관리자만 확인이 가능 )

```
col username for a15
```

```
col default_tablespace for a15
```

```
select username, default_tablespace
```

```
from dba_users
```

```
order by username ;
```

# 로그인 권한 부여

카페 452, 827

- grant 명령으로 시스템 권한을 부여한다.
- 로그인 가능 = create session 권한이 있다.

## 사용 형식

```
grant 시스템권한1, 시스템권한2  
TO 사용자 | 롤 | public;
```

## 사용 예시

```
-- 로그인 권한이 없기 때문에 로그인이 불가능하다.  
conn bigbang/bigbang  
ERROR:  
ORA-01045: user bigbang lacks  
create session privilege; logon denied
```

경고 : 이제는 ORACLE에 연결되어 있지 않습니다.

```
-- 위의 메시지가 뜨는 이유는 로그인 권한(CREATE  
SESSION)이 없어서 뜨는 메시지이다.
```

## 사용 예시

```
-- 관리자 세션  
-- 빅뱅에게 로그인 권한 부여하기  
grant create session to bigbang ;  
  
-- bigbang 세션 : 로그인 가능함  
conn bigbang/bigbang  
  
show user  
  
select * from tab;
```

# 권한 실습

카페 452, 827

- 빅뱅 사용자에서 테이블 생성 실습을 수행해보도록 한다.

## bigbang 세션

```
create table employees(  
    id varchar2(20),  
    name varchar2(30),  
    salary number(10)
```

```
);
```

```
create table employees(  
*
```

1행에 오류:

ORA-01031: 권한이 불충분합니다

-- 위의 메시지는 테이블 생성

-- 권한이 없어서 뜨는 메시지이다.

## 사용 예시

-- 관리자 세션

-- 관리자는 bigbang에게 권한을 부여해줘야 한다.

grant create table, create sequence to bigbang;

-- bigbang 사용자에게 system 테이블 스페이스 허가권 무제한 주  
기

alter user bigbang

quota unlimited on users ;

-- bigbang 세션 테이블 만들기

```
create table employees(  
    id varchar2(20),  
    name varchar2(30),  
    salary number(10)
```

```
);
```

insert into employees values('hong', '홍길동', 100) ;

insert into employees values('kim', '김철수', 200) ;

insert into employees values('park', '박영희', 300) ;

commit ;

# 권한 실습

카페 452, 827

- 현재 oraman, bigbang 양쪽 모두 employees 테이블을 가지고 있다.

## bigbang 세션

```
-- oraman의 employees 테이블 조회해보기  
select id, name from oraman.employees ;  
*
```

1행에 오류:

ORA-00942: 테이블 또는 뷰가 존재하지 않습니다

-- 위의 메시지는 2가지 경우에 발생한다.

-- 첫 번째 : 오타나 실제 테이블이 존재하지 않는 경우

-- 두 번째 : 권한이 불충분하기 때문

## oraman 세션

```
-- oraman 사용자는 employees 테이블에 대하여  
-- bigbang에게 select 권한을 줘야 한다.
```

```
grant select  
on employees  
to bigbang ;
```

-- bigbang 세션

-- 자기 자신 테이블 조회

```
select id, name from employees ;
```

-- oraman 사용자의 테이블 조회

```
select sabun, name from oraman.employees ;
```



# 권한 실습

카페 452, 827

- 나는 oraman 사용자이다.
- bigbang의 employees 테이블을 조회하려고 한다.
- 어떻게 해야 하는지 조치하세요.
- 최종 확인
  - oraman은 bigbang의 employees 테이블에 대한 select가 가능하다.
  - bigbang는 oraman의 employees 테이블에 대한 select가 가능하다.

# 동의어(Synonym)

카페 2338

용어	설명
동의어	객체에 붙이는 적절한 별칭을 말한다. 사용자 및 객체 이름을 간단하게 부르기 위한 용도로 사용된다. 예시 : tab 는 user_tables의 동의어이다.
특징	애플리케이션 개발시 코딩이 불편, 이해하기 어려움
객체 참조 방법	[사용자명].[객체이름] 예를 들어 oraman 사용자의 employees 테이블은 다음과 같이 접근이 가능하다. oraman.employees (이름이 길다)
사용 문법	create [public] synonym [동의어_이름] for [사용자이름][테이블이름] ; create public synonym emp for hr.employees; select * from emp ;

# 동의어(Synonym)

카페 452, 827

- oraman이 bigbang의 employees를 조회하되, 동의어 bigemp을 생성하여 조회해 보세요.

```
col name for a20
select name, salary from bigbang.employees ;

-- 우선 관리자가 public synonym 권한을 부여한다.
grant create public synonym to oraman;

-- oraman 세션 : bigbang의 employees 조회하기 위하여 시너님 생성
create public synonym bigemp for bigbang.employees ;

-- bigbang의 employees을 시너님으로 조회
select name, salary from bigemp ;
```

# 사용자 변경

카페 452, 827

- 사용자 bigbang이 자신의 비번을 까먹었다.
- 이에 관리자는 비번을 바꿔 주고, bigbang 로그인시 비번을 수정하도록 요구해야 한다.
- 사용자의 비밀 번호를 알려 주고, 사용자는 비밀 번호를 변경해야 한다.

```
-- 관리자가 비번을 변경한다.  
-- 비번은 숫자만으로 구성이 불가능하다.  
-- password expire : 암호 만기 옵션  
alter user bigbang  
identified by a1234  
password expire ;
```

```
-- 바뀐 비번으로 bigbang이 로그인해본다.  
conn bigbang/a1234  
ERROR:  
ORA-28001: the password has expired
```

```
bigbang에 대한 암호를 변경합니다  
새 암호: bigbang  
새 암호 다시 입력: bigbang  
...
```

```
암호가 변경되었습니다  
연결되었습니다.
```

# 사용자 삭제

카페 452, 827

- 실습 사용자 sample를 생성하고 삭제하기 테스트
- cascade 옵션 : 보유하고 있던 객체까지 포함하여 삭제한다.

```
-- 사용자 sample 생성
create user sample
identified by sample
quota unlimited on users
account unlock ;
grant connect, create table to sample ;
```

```
-- 사용자 sample 로 로그인
-- sample이 temp 테이블 생성
-- sample 세션
conn sample/sample ;
drop table temp ;
create table temp(
            id number,
            name varchar2(10)
);
insert into temp values(1, '호호') ;
insert into temp values(2, '히히') ;
commit ;
```

```
-- 관리자가 sample 사용자 삭제하기
drop user sample ;
*
```

1행에 오류:  
ORA-01940: 현재 접속되어 있는 사용자는 삭제할 수 없습니다

```
-- sample 사용자 종료하시고 다시 수행
drop user sample ;
*
```

1행에 오류:  
ORA-01922: 'SAMPLE'(을)를 삭제하려면 CASCADE를 지정하여야 합니다

```
-- 무시하고 삭제하겠다.
-- 사용자가 보유한 모든 object를 그냥 지워 버리겠다.
drop user sample cascade ;
```

# 권한(Privilege)

카페 2339

- 사용자에게 대하여 특정 sql 문장이나 객체에 대하여 접근/실행 등을 수행할 수 있는 능력을 의미한다 .
- SQL 구문 등에 대한 사용상의 제한을 말한다.
- 예를 들어서 사용자에게 특정 객체에만 접근 가능하게 한다./못한다.

권한	설명
시스템 권한	데이터 베이스에 액세스하기 위한 권한 create table : 테이블 생성이 가능하다 create session : 접속이 가능한 권한이다.
객체 권한	객체들에 대한 조작에 관한 권한 사용자 oraman은 bigbang 사용자의 employees 테이블에 대한 조회(select) 권한을 가지고 있다.

# 시스템 권한

카페 2339

- 시스템 권한은 100개 이상의 권한 (관리자 접속 후 아래 쿼리 수행 바람)
- 시스템 권한 조회하기
  - select privilege from role\_sys\_privs
  - where role='DBA' order by privilege ;
- 예시 : 사용자 생성, 테이블 제거, 테이블 백업 등등

권한	설명
create user	Oracle 사용자 생성
create any table	나 자신 뿐만 아니라 다른사람의 스키마 영역까지 생성 가능
create session	데이터베이스 접근 권한
create view	뷰 생성 권한
create sequence	시퀀스 생성 권한
create procedure	프로시저 생성 권한
drop user;	사용자 제거하기 권한

# 시스템 권한

카페 2339

## 사용 형식

```
grant 시스템권한1[, 시스템권한2, ...]  
to 사용자[, 사용자 | 롤 |public ];
```

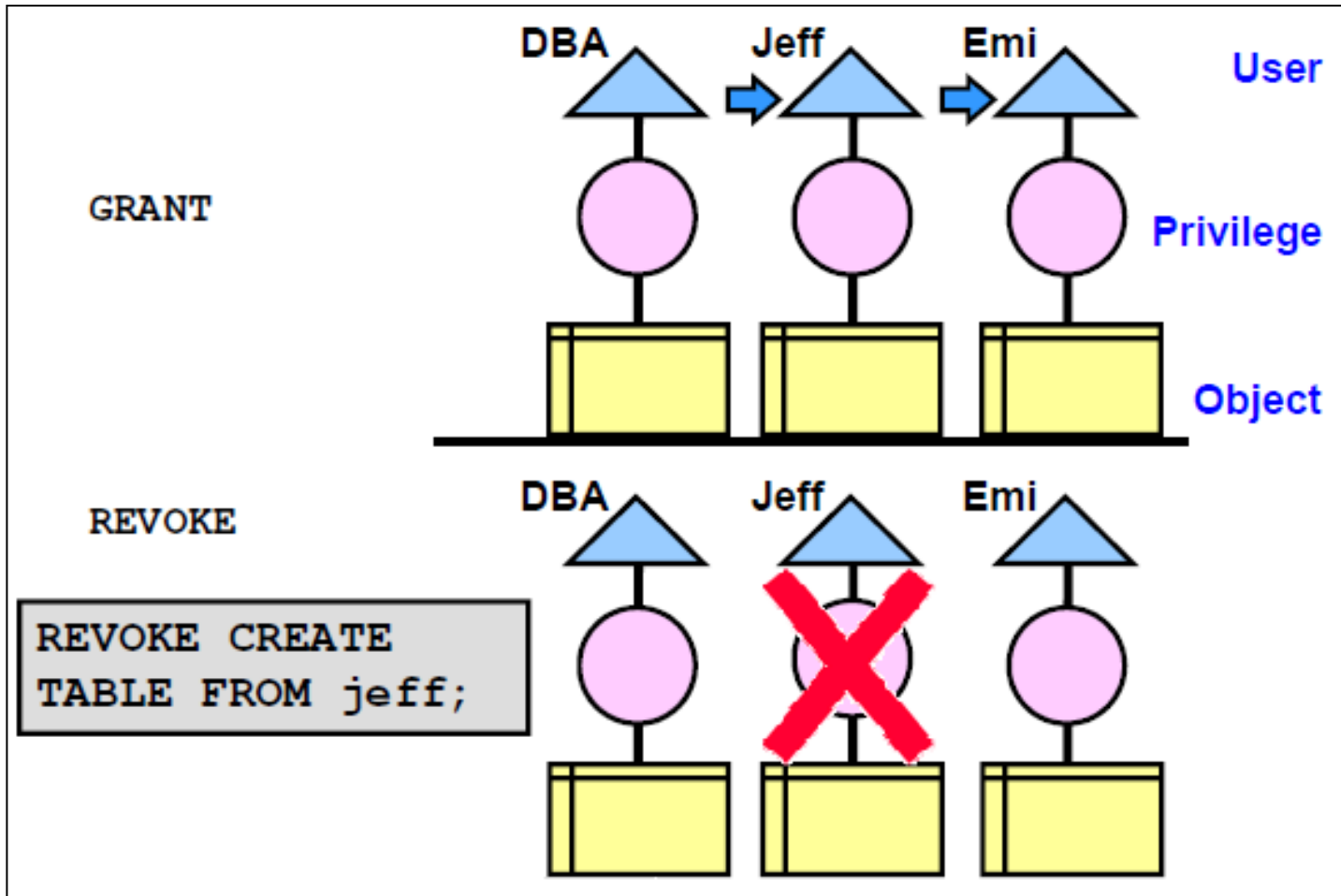
## 사용 예시

```
conn / as sysdba  
grant create session, create table,  
  
create sequence, create view to scott, hr, oraman;  
  
grant create session, create table to bigbang;  
  
-- bigbang 및 oraman 유저로 각각 접속하여 아래 쿼리를 수행해본다.  
  
-- 내가 부여 받은 시스템 권한 보기  
-- sys_privs : System Privilege들  
col privilege for a25  
select * from user_sys_privs;
```



# 시스템 권한 제거(with admin option)

카페 2339



# 시스템 권한 제거(with admin option)

카페 2340

관리자가 oraman에게 create table 생성 권한을 부여하면서 with admin option도 함께 부여한다.

```
grant create table to oraman  
with admin option ;
```

-- oraman 세션 : 테이블 생성이 가능하다  
create table test2(id varchar2(10));

아울러 다른 이에게 create table 생성 권한을 부여할 수 있다.

```
grant create table to bigbang ;
```

-- bigbang 세션 : 테이블 생성이 가능하다.  
create table temp(id varchar2(10));

-- 관리자가 oraman의 create table 권한 박탈  
revoke resource from oraman ;  
revoke create table from oraman ;

-- oraman 세션 : 다음 문장이 실행되지 않는 이유를 설명하시오.

```
create table test3(id varchar2(10));
```

-- bigbang 세션 : 다음 문장은 실행이 되나요?  
create table temp2(id varchar2(10));

# 객체 권한

카페 2339

- 객체 소유자는 객체에 대한 모든 권한을 보유하고 있다.
- 다른 이에게 권한을 줄 수 있다.

## 사용 형식

```
grant 객체권한1[(컬럼)] | all  
on 객체이름  
to 사용자 | 롤 | public;
```

## 사용 예시

```
-- public 옵션  
예를 들어서 table에 대하여 all이라고 한다면 테이블  
에 부여 가능한 모든 권한들을 의미한다.  
public : every one을 의미한다.
```

```
-- 모든 이에게 employees 테이블에 대한 모든 권한  
을 부여하라.  
grant all on employees to public ;
```

# 객체 권한

카페 2340

## 사용 예시

```
-- 객체 권한
-- bigbang의 테이블 employees에 대하여 oraman
이 수정 가능하도록 권한 부여하세요.
-- 단, salary 컬럼만 수정이 가능하도록 한다.
-- bigbang 세션
grant update(salary) on employees
to oraman ;

-- oraman 세션
col name for a15
select * from bigbang.employees ;

update bigbang.employees set salary = 999 ;
commit ;
```

## 사용 예시

```
-- bigbang 세션 : 변경된 데이터를 조회하게 된다.
col name for a15
select * from employees ;
```

```
-- 내가 부여해준 객체 권한 보기
col grantee for a10
col table_name for a10
col column_name for a10
col grantor for a10
col privilege for a10
```

```
select * from user_col_privs_made ;
```

GRANTEE	TABLE_NAME	COLUMN_NAME
GRANTOR	PRIVILEGE	GRANTA

ORAMAN	EMPLOYEES	SALARY
BIGBANG	UPDATE	NO

# 객체의 종류에 따른 권한

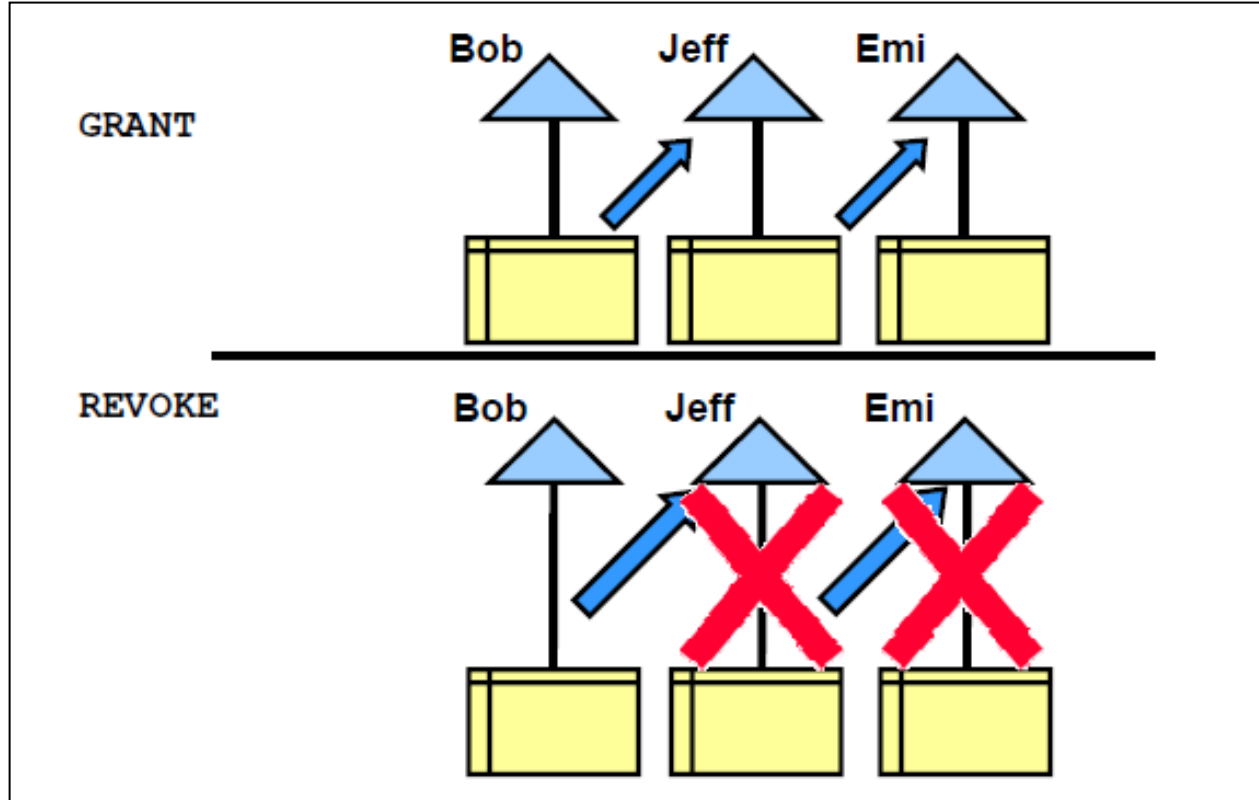
카페 2339

- 객체 권한은 종류에 따라서 부여할 수 있는 권한이 정해진 것도 있다.
- 예를 들어서 시퀀스(sequence) 객체는 Alter(변경)과 Select 권한을 부여할 수 있다.
- 또한 프로시저는 실행(execute) 권한만 가지고 있다.

Object Privilege	Table	View	Sequence	Procedure
Alter	√		√	
Delete	√	√		
Execute				√
Index	√			
Insert	√	√		
References	√			
Select	√	√	√	
Update	√	√		

# 객체 권한 제거 (with grant option)

카페 2339



# 객체 권한 제거

카페 2340

- 객체에 대한 권한을 제거한다.
- REVOKE 명령으로 객체 권한을 제거한다.

## 사용 형식

```
revoke 객체권한1, 객체권한2  
on 객체이름  
from 사용자 | 롤 | public ;
```

## 사용 예시

```
conn / as sysdba  
  
grant create session  
to public;  
  
conn hr/hr  
grant select, insert  
on departments  
to scott ;
```

## 사용 예시

```
conn / as sysdba  
  
revoke create session  
from public ;  
  
conn hr/hr  
  
revoke select, insert  
on departments  
from scott ;
```

# 롤 (role)

카페 2339, 2340

- 여러 개의 권한을 하나로 묶어 놓은 집합체(collection, bottle) 개념으로 보면 된다.
- 롤의 특징
  - 여러 개의 권한을 묶어 두므로 관리가 용이하다.
  - 추가와 삭제를 수행하게 되면 동적으로 사용자에게 반영이 된다.

## 사용 형식

```
-- 사용자 생성 : kara
-- 롤 생성 : myrole
create role myrole

-- 롤 myrole에게 create session, create table,
-- create sequence 권한 부여
grant create session, create table,
create sequence to myrole ;

-- 해당 롤을 kara에게 부여
grant myrole to kara ;

-- 롤이 가지고 있는 모든 권한 보기
select *from role_sys_privs
where role='MYROLE';
```

## 사용 예시

```
-- kara로 접속하셔서 테스트 하세요.
create table sample(
            id number,
            name varchar2(10)
);

create sequence myseq ;

insert into sample
values(myseq.nextval, '김철수') ;
commit ;

select * from sample ;
```



# 미리 정의되어 있는 Role

카페 2339

- Oracle에서는 다음과 같이 미리 만들어 놓은 role들이 존재한다
  - `select count(*) from role_sys_privs where role='resource';`
  - `grant connect, resource to system;`

role 이름	설명
connect	create session, em 에서 생성시 자동으로 부여됨
resource	create cluster, create indextype, create operator, create procedure, create sequence, create table, create trigger, create type, unlimited tablespace
scheduler_admin	create any job, create external job, create job, execute any class, execute any program, manage scheduler
dba	most system privileges, several other roles do not grant to nonadministrators
select_catalog_role	no system privileges, but hs_admin_role and over 1,700 object privileges on the data dictionary

# 딕셔너리 뷰

- 객체 생성을 하게 되면 데이터 사전에 등록이 된다.
- 이 데이터 사전을 관리하기 위한 사전을 dictionary(줄여서 dict)라고 부른다.

## 사용 형식

```
-- 사전도 테이블이므로 desc 가능하다
desc dict
```

이름	널?	유형
----	----	----

TABLE_NAME		VARCHAR2(30) <-- 등록된 사전 이름
COMMENTS		VARCHAR2(4000) <-- 간단한 코멘트

```
-- 예를 들어서 [권한] 관련 사전 조회하기
```

```
col table_name for a20
```

```
col comments for a45
```

```
select * from dict where table_name like '%PRIVS%' ;
```

# 딕셔너리 뷰

- `select * from dict where table_name like '%PRIVS%'`
- PRIVS 단어 대신 다음 단어로 대체해서 확인
  - `_COL_`(컬럼), `_TAB_`(테이블), `PRIVS`(권한\_PRIVILEGE), `ROLE`(롤)
  - `_IND_`(인덱스), `USER` : 자기 자신

딕셔너리 이름	설명
role_sys_privs	롤에 부여된 시스템 권한 정보
role_tab_privs	롤에 부여된 테이블 관련권한 정보
user_role_privs	접근 가능한 롤의 정보
user_tab_privs_made	해당 사용자 소유의 오브젝트에 대한 오브젝트 권한 정보
user_tab_privs_recd	사용자에게 부여된 오브젝트 권한 정보
user_col_privs_made	사용자 소유의 오브젝트 중에서 컬럼에 부여된 오브젝트 권한 정보
user_col_privs_recd	사용자에게 부여된 특정 컬럼에 대한 오브젝트 권한 정보

# Oracle(SQL)

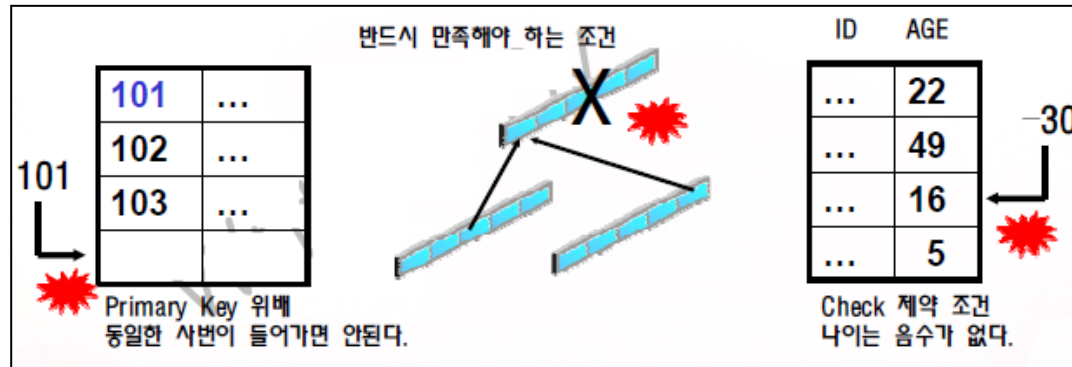
제약 조건 (Constraint)



# 제약 조건

카페

- 무결성 : 데이터베이스 내의 데이터에 대한 정확성을 유지하는 것을 의미한다.
- 정의 : 무결성을 유지하도록 컬럼에 제약(제한 사항)을 주는 것을 의미한다.
- 제약 조건 : 원하지 않는 데이터가 입력/수정되는 것을 방지하기 위함이다.



not null	컬럼에 null 값을 허용하지 않음(null 값 불허)
unique	중복된 값을 허용하지 않는다. 널 값은 유일한 조건인지 체크하지 않는다.(2 개 이상 입력 가능 )
primary key	not null + unique
foreign key	참조되는 테이블의 컬럼 값이 존재하면 허용
check	저장 가능한 값의 범위 또는 조건을 지정하여 설정 값만 허용 예시 : 급여 컬럼은 항상 0 이상이다. 성별 컬럼은 [남] 또는 [여]가 되어야 한다.

# not null

카페

- 사원 번호, 사원 이름을 not null으로 설정

## 사용 예시

관리자 세션 : oraman 사용자에게 system 테이블스페이스 공간을 무제한 제공

```
alter user oraman quota unlimited on users ;
```

-- 사용자 oraman

conn oraman/oracle

```
create table emp01(
```

```
    empno number,
```

```
    ename varchar2(10),
```

```
    job varchar2(9),
```

```
    deptno number(4)
```

```
);
```

-- not null을 부여하지 않았으므로, null 값이 들어갈 수 있다.

```
insert into emp01 values(null, null, 'salesman', 30) ;
```

```
commit ;
```

```
select * from emp01 ;
```

EMPNO	ENAME	JOB	DEPTNO
-------	-------	-----	--------

-----

		salesman	30
--	--	----------	----

## 사용 예시

-- 컬럼에 not null : 반드시 넣어 줘야 하는 컬럼으로 필수 사항을 의미한다.

```
create table emp02(
```

```
    empno number not null,
```

```
    ename varchar2(10) not null,
```

```
    job varchar2(9),
```

```
    deptno number(4)
```

```
);
```

```
insert into emp02 values(null, null, 'salesman', 30) ;
```

\*

1행에 오류:

ORA-01400: NULL을 ("ORAMAN"."EMP02"."EMPNO") 안에 삽입할 수 없습니다

-- 해결 방법

1) 해당 컬럼에 데이터를 반드시 넣어 준다.

2) 테이블 구조를 변경한다.

# unique

카페

- 특정 컬럼에 대하여 자료가 중복이 되지 않도록 설정.
- Unique는 내부적으로 인덱스를 만들어 준다.

## 사용 예시

```
-- unique : 입력되는 데이터에 대하여 중복이 되면 안된다.  
-- 단, null이 들어 오면 신경 쓰지 않겠다.  
-- 예를 들어서, 주민번호와 email은 unique 대상이다.
```

```
-- 사원 번호를 유일한 키로 지정한다.  
create table emp03(  
    empno number unique,  
    ename varchar2(10) not null,  
    job varchar2(9),  
    deptno number(4)
```

```
);  
insert into emp03  
values(7499,'allen', 'SALESMAN', 30);
```

```
--여기 까지는 별 문제 없음  
select * from emp03;
```

```
EMPNO ENAME JOB DEPTNO
```

```
-----  
7499 ALLEN SALESMAN 30
```

## 사용 예시

```
-- 동일한 사번을 다시 입력  
insert into EMP03  
values(7499,'JONES','MANAGER', 20);  
*
```

```
1행에 오류:  
ORA-00001: 무결성 제약 조건  
(CONSTSYS_C005488 ) 에 위배됩니다
```

```
--하지만 null 값은 중복이 되게 저장 가능  
--즉, Unique 는 값이 중복이 되지 않음을 의미
```

```
--아래의 예를 보라  
insert into EMP03  
values(null,'JONES','MANAGER', 20);  
insert into EMP03  
values(null,'JONES','SALESMAN', 10);  
select * from EMP03;
```

# unique

- Unique는 내부적으로 인덱스를 만들어 준다.

## 사용 예시

-- 인덱스의 이름과 속한 TBS 확인하기

```
select table_name, index_name, tablespace_name from user_indexes  
where table_name='EMP03';
```

```
CONST> @const  
table_name의 값을 입력하십시오 : emp03
```

TABLE_NAME	CONSTRAINT_NAME	C	STATUS	COLUMN_NAME	SEARCH_COND
EMP03	SYS_C006311	C	ENABLED	ENAME	"ENAME" IS not null
EMP03	SYS_C006312	U	ENABLED	EMPNO	



# USER\_CONSTRAINTS

카페

- 제약 조건을 확인하기 위한 딕셔너리 뷰

컬럼 이름	설명
constraint_name	제약 조건 이름
constraint_type	제약 조건의 유형
table_name	제약 조건이 속한 테이블 이름

CONSTRAINT_TYPE	의미
P	Primary Key
R	Foreign Key
U	Unique
C	Check, not null

## 스크립트 (const.sql 스크립트 파일 내용)

```
col column_name format a15  
col table_name format a15  
col constraint_name format a20
```

```
select t.table_name, t.constraint_name, t.constraint_type, t.status, c.column_name,  
t.search_condition from user_constraints t, user_cons_columns c  
where t.table_name = c.table_name  
and t.constraint_name = c.constraint_name  
and t.table_name = upper('&table_name') ;
```

# primary key

카페

- UNIQUE + not null 조건

## 사용 예시

```
create table emp04(  
    empno number primary key,  
    ename varchar2(10) not null,  
    job varchar2(9),  
    deptno number(4)  
);  
  
insert into emp04  
values(7499, 'allen', 'salesman', 20);  
  
-- 여기까지는 별 문제 없음  
select * from emp04;  
values(null, null, 'salesman', 30);
```

## 사용 예시

```
-- 동일한 사번을 다시 입력  
-- unique 조건에 위배  
  
insert into emp04  
values(7499, 'JONES', 'MANAGER', 20);  
*  
1행에 오류:  
ORA-00001: 무결성 제약 조건  
(CONSTSYS_C005492 ) 에 위배됩니다  
  
--스크립트로 확인  
@const.sql 수행 후 [emp04 입력]
```

# primary key

- UNIQUE + not null 조건

## 사용 예시

```
--not null 조건 위배  
insert into EMP04  
values(null, 'JONES', 'MANAGER', 20);  
*
```

2행에 오류:

ORA-01400: null 을 ("CONST""EMP04""EMPNO") 안에 삽입할 수 없습니다  
UNIQUE + not null 조건

--인덱스의 이름과 속한 TBS 확인하기

```
select table_name, index_name, tablespace_name from user_indexes  
where table_name='EMP04';
```

```
TABLE_NAME INDEX_NAME TABLESPACE_NAME
```

```
-----  
EMP04 SYS_C006314 USERS
```

```
CONST> @const
```

```
table_name의 값을 입력하십시오 : emp04
```

# foreign key

카페

- 외래 키 제약 조건 설정

## 사용 예시

```
-- DEPT 테이블 : master 테이블

-- FOREIGN KEY 제약 조건 설정하기
create table emp05(
    empno number primary key,
    ename varchar2(10) not null,
    job varchar2(9),
    deptno number(4)
    references dept(deptno)
);
```

## 사용 예시

```
--여기까지 별 문제 없음
insert into emp05
values(7499, 'ALLEN', 'SALESMAN', 50);

select * from emp05 ;

-- 50 번 부서는 없다.
insert into emp05
values(7500,'ALLEN','SALESMAN', 50);
*
1행에 오류:
ORA-02291: 무결성 제약 조건
(CONSTEMP05_DEPTNO_FK) 이 위배되었습니다.
부모키가 없습니다

@CONST.SQL
```

# check

카페

- 특정 컬럼에 대한 값의 제약 조건

## 사용 예시

```
create table emp06(  
    empno number primary key,  
    ename varchar2(10) not null,  
    gender varchar2(1)  
    check(gender in('m','f'))  
);  
  
--여기까지 별 문제 없음  
insert into emp06  
values(7499,'ALLEN','M');  
  
select * from emp06;
```

## 사용 예시

```
-- 존재하지 않는 값 입력  
insert into emp06  
values(7500, 'ALLEN', 'S');  
*  
1행에 오류:  
ORA-02290: 체크 제약 조건  
(CONSTSYS_C005499) 이 위배되었습니다.  
  
-- @CONST.SQL 파일 수정  
-- select 절 마지막에 SEARCH_CONDITION  
-- 컬럼 추가 후 저장  
@CONST.SQL
```

# user\_objects

- 사용자가 보유하고 있는 모든 오브젝트를 보기 위한 디렉터리 뷰

## 사용 예시

-- 오브젝트의 이름과 종류 조회하기

```
col object_name for a15
```

```
col object_type for a15
```

```
select object_name, object_type  
from user_objects ;
```

```
OBJECT_NAME OBJECT_TYPE
```

```
-----
```

```
EMP02          TABLE
```

```
EMP03          TABLE
```

```
EMP01          TABLE
```

```
SYS_C006312    INDEX
```

```
SYS_C006314    INDEX
```

결과를 보게 되면 테이블 3개와 인덱스 2개로 구성이 되어 있다.

# 제약 조건의 이름

카페

- 미지정시 Oracle이 자동으로 명명 : SYS\_Cxx
- 권장 사항 : 사용자가 명시적으로 지정 요망
  - 차후 문제 발생시 유지 보수 용이성을 위해서 필요하다
  - 시스템이 명명한 것은 디셔너리 뷰를 다시 봐야 하는 문제점이 있음
- 사용 형식 :
  - 컬럼\_이름 데이터\_타입 CONSTRAINT 제약\_조건\_이름 제약\_조건\_타입
- 제약 조건 명명 규칙
  - 테이블이름\_컬럼이름\_제약조건유형
- 제약 조건 유형
  - PK / UK / NN / CK / FK 명명 예시
- 명명 예시
  - EMP05\_EMPNO\_PK / EMP01\_LOC\_NN / EMP02\_GENDER\_CK

# 제약 조건의 이름 변경

카페

- 이전에 만든 테이블의 제약 조건의 이름을 변경해보자

## 사용 예시

```
alter table 테이블이름  
rename constraint 이전제약조건이름 to 신규제약조건이름 ;
```

-- emp02 ~emp06까지 모두 변경하십시오.(SYS\_Cxx 없애기)

-- @const2.sql를 사용하십시오

TABLE_N	CONSTRAINT_N	C	STATUS	COLUMN_	SEARCH_CONDITION
---------	--------------	---	--------	---------	------------------

EMP02	SYS_C006309	C	ENABLED	EMPNO	"EMPNO"IS not null (하단 예시 참고)
EMP02	SYS_C006310	C	ENABLED	ENAME	"ENAME"IS not null
EMP03	SYS_C006311	C	ENABLED	ENAME	"ENAME"IS not null
EMP03	SYS_C006312	U	ENABLED	EMPNO	
EMP04	SYS_C006313	C	ENABLED	ENAME	"ENAME"IS not null
EMP04	SYS_C006314	P	ENABLED	EMPNO	

```
alter table emp02  
rename constraint SYS_C006309 to EMP02_EMPNO_NN;
```



# 제약 조건의 명시적 지정

## 사용 예시

```
drop table emp05 purge ;
```

```
create table emp05(  
    empno number(4) constraint emp05_empno_pk primary key,  
    ename varchar2(10) constraint emp05_ename_nn not null,  
    job varchar2(9) constraint emp05_job_uk unique,  
    deptno number(4) constraint emp05_deptno_fk references dept(deptno)  
);
```

```
insert into emp05 values(1234, '김철수', '판매원', 10);  
commit ;
```

@const 수행 후 emp05 입력 후 엔터

TABLE_N	CONSTRAINT_N	CO	STATUS	COLUMN_	SEARCH_CONDITION
EMP05	EMP05_ENAME_NN	C	ENABLED	ENAME	"ENAME" IS NOT NULL
EMP05	EMP05_EMPNO_PK	P	ENABLED	EMPNO	
EMP05	EMP05_JOB_UK	U	ENABLED	JOB	
EMP05	EMP05_DEPTNO_FK	R	ENABLED	DEPTNO	

# 제약 조건의 명시적 지정

컬럼 이름	제약 조건	사용자가 지정한 제약 조건명
EMPNO	PRIMARY KEY	EMP05_EMPNO_PK
ENAME	not null	EMP05_ENAME_NN
JOB	UNIQUE	EMP05_JOB_UK
DEPTNO	REFERENCES(FOREIGN KEY)	EMP05_DEPTNO_FK

## 사용 예시

COL SEARCH\_CONDITION FOR A15  
@const.sql 수행 후 EMP05 엔터

```
-- not null 조건 위반
insert into EMP05
values(7499, null, 'MANAGER', 50);
*
```

2 행에 오류:  
ORA-01400: null 을 ( "CONST"."EMP05"."ENAME")  
안에 삽입할 수 없습니다.

## 사용 예시

```
-- PRIMARY KEY 조건 위반
insert into emp05
values(7499,'ALLEN','SALESMAN', 50); *
1 행에 오류:
ORA-00001: 무결성 제약 조건
(CONSTEMP05_EMPNO_PK)에 위반됩니다
```

```
insert into emp05
values(7500,'ALLEN','SALESMAN', 50);
```

# 제약 조건 지정 방식

방식	설명
컬럼 레벨	<p>컬럼 정의한 바로 직후에 이어서 후에 제약 조건을 지정하는 방식</p> <p>반드시 컬럼 레벨 단위로 지정해야 하는 것</p> <ul style="list-style-type: none"> <li>* not null 제약 조건</li> </ul>
테이블 레벨	<p>테이블 정의 마무리 직전에 제약 조건을 정의하는 방식</p> <p>반드시 테이블 레벨 단위로 지정해야 하는 것</p> <ul style="list-style-type: none"> <li>* 복합 키로 기본 키를 지정하는 경우</li> <li>* ALTER TABLE 명령어를 사용하는 경우</li> </ul>

## 사용 형식

-- 테이블 레벨의 지정 형식

```
create TABLE 테이블이름 (
    컬럼1 데이터타입 1,
    컬럼2 데이터타입 2,
    ...
    [CONSTRAINT 이름] 타입 ( 컬럼이름 )
)
```

# 제약 조건 지정하기

## 사용 예시

```
-- 컬럼 레벨에서 지정 예시
drop table emp04 purge ;

create table emp041(
    empno number(4) primary key,
    ename varchar2(10) not null,
    job varchar2(9) unique,
    deptno number(4) references dept(deptno)
);
```

## 사용 예시

```
-- 테이블 레벨에서 지정 예시
create table emp042(
    empno number(4),
    ename varchar2(10) not null,
    job varchar2(9),
    deptno number(4),
    primary key(empno),
    unique(job),
    foreign key(deptno) references dept(deptno)
);
```

# 복합 키 제약 조건 지정

카페 438

## 사용 예시

- 복합 : composite, 2개 이상의 집합을 의미한다.
- 복합(composite) 키는 무조건 테이블 레벨에서 지정해야 한다.
- composite primary key : 컬럼 2개를 가지고 만드는 primary key이다.
- 필요한 경우 : 2개 이상의 컬럼이 서로 합쳐져서 시너지 효과를 보는 경우에 많이 사용

```
create table member(  
    name varchar2(20),  
    hphone varchar2(15),  
    address varchar2(50),  
    constraint member_composite primary key(name, hphone)  
);
```

- 복합 키 설정하기 : 2개 이상의 컬럼을 가지고 제약 조건 설정하기
- user\_cons\_columns : 복합키의 컬럼에 대한 정보를 보여 주는 디렉터리 뷰
- POSITION 컬럼에 유의하여 확인 요망

```
col owner for a8  
col table_name for a10  
col column_name for a15  
select * from user_cons_columns  
where table_name = 'MEMBER01';  
OWNER CONSTRAINT_N TABLE_NAME COLUMN_NAME POSITION
```

```
-----  
CONST MEMBER01_COMBO_PKMEMBER01 NAME 1  
CONST MEMBER01_COMBO_PKMEMBER01 HPHONE 2
```

- 특정 테이블의 primary key 갯수
- $0 \leq \text{PK갯수} \leq 1$
- 단일 키 : 컬럼 1 개로 구성
- 복합키 : 컬럼 2 개 이상으로 구성

# 제약 조건의 변경

- alter table 테이블이름... 구문을 이용하여 제약 조건을 변경한다.

방식	설명
추가하기	alter table 테이블이름 add [constraint 제약_조건_이름] 제약 조건_타입 ( 컬럼이름 ) ;
추가하기((pk)	primary key는 예외적으로 다음과 같이 수행해도 동일한 결과가 도출된다. alter table 테이블이름 add primary key(empno);
추가하기(not null)	alter table 테이블이름 modify ( 컬럼이름 constraint 제약_조건_이름 not null) ;
삭제하기	alter table 테이블이름 drop [constraint 제약_조건_이름] ;
삭제하기(pk)	primary key는 예외적으로 다음과 같이 수행해도 동일한 결과가 도출된다. alter table 테이블이름 drop primary key ;

# 제약 조건의 변경

## 사용 예시

```
--실습용 테이블 생성
drop table emp01;
create table emp01(
    empno number(4),
    ename varchar2(10),
    job varchar2(9),
    deptno number(4)
);
@const.sql

-- 기본 키 추가
alter table emp01 add primary key(empno);

-- 외래 키 추가
alter table emp01
add constraint emp01_deptno_fk foreign
key(deptno)
references dept(deptno) ;

@const.sql
```

## 사용 예시

```
-- not null 추가
alter table emp01 modify ename not null;

alter table emp01 modify job constraint
emo01_job_nn not null;
@const.sql

-- 다음 내용은 오류가 발생한다
insert into emp05
values(7499, 'allen', 'sales', 30)

-- 실습자에 따라 다를 수 있음
-- 위에서 확인한 기본 키 삭제
alter table emp05
drop constraint emp05_empno_pk ;

-- 기본 키는 다음과 같이 삭제할 수도 있다.
-- alter table emp5 drop primary key;
insert into emp05 values(7499,'allen','sales',
30);
```

# 제약 조건의 변경

## 사용 예시

-- 사원 이름에 null 이 입력 가능하도록 not null 조건 제거

```
alter table emp05
```

```
drop constraint emp05_ename_nn ;
```

```
insert into emp05
```

```
values(7499, null, 'analyst', 30);
```

```
@const.sql
```

-- 나머지 제약 조건도 삭제해보세요

-- 지금까지 실습한 emp01 ~ emp06 테이블 모두의 제약 조건을 삭제해 보세요.

-- const2.sql 스크립트를 같이 사용하세요.



# Oracle(SQL)

뷰(View)



# 뷰(View)

카페 342

항목	설명
정의	다른 테이블이나 뷰에서 파생된 논리적인 가상 (virtual) 테이블
특징	물리적인 저장 공간이 존재하지 않는다. 쿼리(조회를 위한 select 문) 문장을 저장하고 있는 객체이다. 제한성 : 직접적인 데이터 액세스 제한 용이성 : 복잡한 질의를 쉽게 만들기 위함 한 개의 테이블에 뷰의 개수는 제약이 없다.
관련 권한	create VIEW 예시 : 사용자에게 뷰 생성 권한 부여하기 grant create view to oraman;
데이터 사전	USER_VIEWS 사건의 TEXT 컬럼을 보면 확인이 가능

	단순 뷰	복합 뷰
테이블 개수	1개	n개( $n \geq 2$ )
그룹 함수 사용 가능 여부	불가능	가능
distinct 사용 가능 여부	불가능	가능
dml 사용 가능 여부	가능	불가능

# DML 사용이 불가능한 View

카페

- DML 사용이 불가능한 경우는 다음과 같은 경우이다.
  - not null 제약 조건의 컬럼이 뷰에 포함 되지 않는 경우
  - 가상컬럼 (SQL \*12)인 경우
  - distinct을 포함하는 경우
  - 그룹 함수나 group by절을 포함한 경우

# 뷰 관련 디렉터리

카페

- user\_views의 text 컬럼에 sql 문장이 저장되어 있다.
- 주의 : 객체 이름은 모두 대문자로 처리한다.

## 사용 예시

```
select text from user_views  
where view_name = 'VIEW01';
```

TEXT

---

```
select SABUN, NAME, SALARY, MANAGERID  
from viewexam  
where managerid = 1
```

```
select text from user_views  
where view_name = 'VIEW02';
```

TEXT

---

```
select name, salary from viewexam  
where salary >= 400
```

## 사용 예시

```
select text from user_views  
where view_name = 'VIEW03';
```

TEXT

---

```
select managerid, sum(salary) sumsal  
from viewexam  
group by managerid
```

# 뷰 실습 사전 준비

카페

- drop table viewexam purge ;
- drop sequence viewseq ;
- create table viewexam(
  - sabun number(6) primary key,
  - name varchar2(20) not null,
  - salary number(8,2) check(salary >= 100),
  - managerid number
- );
- create sequence viewseq start with 1 increment by 1 maxvalue 100000;
- insert into viewexam values( viewseq.nextval, '김철수', 300, 1 ) ;
- insert into viewexam values( viewseq.nextval, 김철수, 300, 1 ) ;
- insert into viewexam values( viewseq.nextval, '박영희', 500, 1 ) ;
- insert into viewexam values( viewseq.nextval, '바둑이', 100, 2 ) ;
- insert into viewexam values( viewseq.nextval, '땡칠이', 400, 2 ) ;
- commit ;
- select \* from viewexam ;
- sabun name salarymanagerid
- -----
- 1 김철수 300 1
- 2 박영희500 1
- 3 바둑이 100 2
- 4 땡칠이 400 2

# 뷰 실습

카페

## 사용 예시

매니저 아이디가 1번인 직원들의 정보를 조회하는 뷰  
create or replace view view01  
as select \* from viewexam  
where managerid = 1 ;

select \* from view01 ;  
SABUN NAME SALARY MANAGERID

---

1	김철수	300	1
2	박영희	500	1

급여가 400이상인 직원들의 이름/급여 조회하는 뷰  
create or replace view view02  
as  
select name, salary from viewexam  
where salary >= 400 ;

select \* from view02 ;  
NAME SALARY

---

박영희	500
땡칠이	400

## 사용 예시

매니저 별 직원의 급여의 총합을 구해주는 뷰  
create or replace view view03  
as  
select managerid, sum(salary) sumsal  
from viewexam  
group by managerid ;

select \* from view03 ;

MANAGERID SUMSAL

---

1	800
2	500

# 단순 뷰에 대한 DML

카페

- view는 제한적으로 DML이 가능하다.

## 사용 예시

view01 뷰를 이용하여 테이블에 데이터를 INSERT하시오  
insert into view01 values( viewseqnextval,'홍길동', 800, 1 );

select \* from view01 ;  
SABUN NAME SALARYMANAGERID

-----  
1 김철수 300 1  
2 박영희500 1  
5 홍길동 800 1

view02 뷰를 이용하여 테이블에 데이터를 INSERT하시오  
insert into view02 values( viewseqnextval,'홍길동', 800, 1 );  
\*

1 행에 오류:  
ORA-00913: 값의 수가 너무 많습니다.

원인 : 실제 명시되어 있는 컬럼은 2개 (name, salary)인데,  
4개의 컬럼으로 데이터를 INSERT하려고 하고 있다.

## 사용 예시

insert into view02 values('강감찬', 400 );  
\*

1 행에 오류:  
ORA-01400: null을  
( "ORAMAN""VIEWEXAM""SABUN")  
안에 삽입할 수 없습니다.

원인 : sabun 컬럼은 primary key라서 널 값을  
INSERT하지 못한다.

OX 문제 : view02를 사용하게 되면 viewexam  
테이블에 데이터를 INSERT할 수 있다.

문제 : view03를 사용하게 되면 viewexam 테이블에  
데이터를 INSERT할 수 있나요?

# 복합 뷰

- 자주 조인되는 테이블을 복합 뷰로 구현

## 사용 예시

조인을 이용한 뷰만들기

```
create or replace view complex  
as  
select name, d.deptno, dname  
from employees e, depts d  
where e.deptno =d.deptno  
and d.deptno = 20 ;
```

```
select *from complex ;
```



# with check option

- 조건(where 절)으로 지정한 컬럼을 변경 불가

## 사용 예시

```
create or replace view chkoption  
as  
select * from viewexam  
where managerid = 1  
with check option ;
```

```
select * from chkoption ;
```

SABUN	NAME	SALARY	MANAGERID
-------	------	--------	-----------

1	김철수	300	1
2	박영희	500	1
5	홍길동	800	1

## 사용 예시

```
update chkoption set managerid = 2  
where sabun = 2 ;  
*
```

1행에 오류:

ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배됩니다.

원인 : 사번 2번은 매니저 id가 1이므로 with check option의 지배를 받고 있으므로 업데이트를 수행할 수 없다.

하지만 다음 문장은 with check option과 무관하므로 업데이트가 가능하다.

```
update chkoption set salary = 1000  
where sabun = 2 ;
```

# with read only

- 조건(where 절)으로 지정한 컬럼을 변경 불가

## 사용 예시

```
create or replace view readonly  
as  
select * from viewexam  
where managerid = 1  
with read only ;
```

```
select * from readonly ;
```

```
update readonly set managerid = 2  
where sabun = 2 ;  
*
```

1 행에 오류:

ORA-01733: 가상 열은 사용할 수 없습니다

# top-n 구문

카페 2346

- top-n 구문이란, 특정 테이블 내에서 랭킹을 이용하여 상위 n(정수)개의 데이터를 추출해 내는 구문을 의미한다.
- from절 내에 subquery가 존재하는 뷰를 inline view라고 한다.
- 이러한 inline view를 이용하여 top-n 기능을 구현할 수 있다.
  - 예시 : 전체 순위 6위부터 10위까지 구하기
- 제한 조건
  - 사용 가능한 비교 연산자는 <, <= 이다.
  - subquery 내에서 order by 구문은 사용 가능하다.
- 필요한 이론
  - rownum 컬럼
    - 데이터가 조회될 때 만들어지는 가상 컬럼
    - 가장 먼저 조회되는 row의 rownum 은 1이다.(순서를 나타내는 컬럼 )

topN\_실습 .txt

# 페이징(paging)

▶ 파일자료실 > 영화

20개 50개 90개

<input type="checkbox"/>	번호	제목	용량	가격	분류	판매자
<input type="checkbox"/>	5363261	<b>[추격자]</b> The.Chaser.2008.720p.BluRay.x264-CiNEFiLE.mkv	0.0G		한국영화	푸니G
<input type="checkbox"/>	5367948	<b>[모니터링]</b> [추억]백재미 요리왕 주성치, 석신	1.4G		코미디	모니터링
<input type="checkbox"/>	5367934	<b>[모니터링]</b> [추억]주성치의 사랑싸움.소자병변	1.4G		코미디	모니터링
<input type="checkbox"/>	5320081	전쟁이들.2012.720p.HDRip-김수호,강혜원,이제훈. (20)	4.0G		최신/미개봉	영향사
<input type="checkbox"/>	5326656	<b>[Bluray.1080p][:-스타치:-]</b> 잔민합화국막의절묘한조화.완자.. (4)	5.4G		최신/미개봉	뷰티모브저
<input type="checkbox"/>	5367939	지마이조2009-전쟁의서막	2.9G		액션	스마일업로드
<input type="checkbox"/>	5367754	2012 반드시 살아야한다 - 실경구 손혜진 HDRip  (3)	3.5G		최신/미개봉	원손그림
<input type="checkbox"/>	5367930	2013년3월7일.아카데미 최고 문제작[ 단 하나의 타겟 ]	2.6G		최신/미개봉	삼쌍
<input type="checkbox"/>	5355635	<b>[핑크곡지]</b> 1080p 초고화질 <b>[ 호빗 ]</b> 반지의 제왕 피터잭슨.. (3)	13.1G		대용량/DVD	핑크곡지
<input type="checkbox"/>	5367928	<b>[모니터링]</b> [추억]주성치레전드.소림축구	1.4G		코미디	모니터링

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

# Oracle(SQL)

인덱스(Index)



# rowid

- Oracle 서버가 자동 관리하는 각 행을 구분 짓기 위한 고유 식별 번호
- 블록간 서로 구분을 위한 유일한 블록 주소 (unique address)를 사용
- 디스크에 10 바이트의 저장 영역이 필요하며, 총 18 문자를 사용하여 표시한다.

- 사용 예시

- select rowid, empno, ename from emp;

- A~Z, a~z, 0~9, / 로 이루어진 64개의 코드 방식

- rowid : 6363 체계

- 6: object number
  - 3 : file number
  - 6: blocknumber
  - 3 : 해당 블록의 행의 위치

Rowid\_실습.txt / ROWID.txt

