-----------------------------------------------------------160801-------------------------------------------------------

http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html

Oracle Database 11g Release 2 에서 윈도우 버전에 맞춰서 다운로드 받는다.

ryon81@gmail.com

Rjwuqjfu18

\*주의사항: 설치 후 복구가 어려움. 실패할 경우 경우에 따라서 윈도우를 재설치 해야 함.

\*사용안할 경우 백그라운드에서 Service되어 무거움. 사용안할 경우 서비스를 사용안함으로 바꾼다.

압축을 다운로드 폴더에 푼다.

2of2 폴더 내에 database\stage\components 로 이동한다. (32bit는 3개 폴더 // 64bit는 4개 폴더 )

1of1 폴더 내에 database\stage\components 로 이동한다. (32bit는 148개 폴더 // 64bit는 164개 폴더)

1of1 폴더 내에 setup 파일을 실행한다.

DOS 실행화면 이후 설치 마법사가 실행한다.

1단계: 정보 이메일 받기 (X) -> NEXT

2단계: 그냥 두기 -> NEXT

3단계: 그냥 두기 -> NEXT

4단계: D 드라이브로 선택한다. (기본설정인 경우 그대로 둔다)

5단계: 관리자 비밀번호 입력한다.

\*수업용은 a로 입력하고 무시한다.

\*데이터베이스 설치한 곳: D:\app\SEC <-- 만약 파일이나 폴더가 이미 설치되어 있는 경우에는 해당 폴더를 삭제해야지만 설치 진행이

가능하다.

6단계: 완료

7단계: 자동설치

\*설치 중 방화벽 차단이 뜨는 경우 엑세스를 허용하면 된다.

8단계: 설치 완료 후 발생하는 팝업창에서 확인을 누르면 설치 종료

시작 -> 모든프로그램 -> Oracle - OraDb11g\_home1 -> 응용 프로그램 개발 ->

SQL plus로 프로그램 실행

\*DOS 창으로 실행됨

연습용 SCOTT 계정추가 필요

계정 추가방법

1. 실행창: cmd 입력하여 dos창을 연다

2. 아래 명령어 입력 <-- system은 관리자명 a 는 관리자비밀번호

sqlplus system/a

3. SQL에 접속한 후 모든 계정정보를 확인한다.

select \* from all\_users;

4. 아래 명령어를 입력하여 "연결되었습니다." 메시지를 확인한다.

conn scott/tiger;

5. 만약 ORA-28000: the account is locked 메시지가 확인되는 경우, 아래와 같이 진행하여 "사용자가 변경되었습니다" 메시지를 확인한

다.

conn system/a;

alter user scott identified by tiger account unlock;

6. 이후에는 4번의 명령어를 다시 입력하여 "연결되었습니다" 메시지를 다시 확인한다.

7. 아래 명령어를 입력하여 scott 계정에 접속한다.

conn scott/tiger;

8. 아래 명령어를 입력하여 현재 설치된 테이블을 볼수 있다.

select \* from tab;

-----------------------------------------------------------160802-------------------------------------------------------

DBMS (Database Management System)

RDBMS (Relational Database Management System) - 가장 일반적인 형태

시장에서는 Oracle, MYSQL 을 주로 사용

이용하기가 쉽고 확장이 쉽다.

첫줄은 컬럼 나머지는 레코드 (데이터 또는 로우라고도 불림)

SQL (Structure Query Language)

SLQplus <-- SQL문법을 이용하여 작성할 수 있는 프로그램

DQL Data Quesry Language

- 데이터 검색할 때 사용

DML Data Manipulation Language

- insert 데이터 삽입에 사용

- update 데이터 수정에 사용

- delete 데이터 삭제에 사용

DDL Data Definition Language

- create 데이터를 넣을 수 있는 테이블으르 생성

- alter 테이블을 수정할 때 사용

- rename 테이블 명을 변경

- truncate 테이블 내용을 제거할 때

- drop - 테이블 자체를 삭제할 때

TCL Transaction Control Language

-

DCL Data Control Language

- grant 권한 부여

- revoke 권한 제거

SQLplus에 접속하기

sqlplus scott/tiger

계정과 암호까지 기억이 나지 않는 경우 아래와 같이 한다. 사용자면 account, 새로운 비번은 tiger

alter user scott identigied by tiger account unlock;

show user 현재 접속한 정보를 보기 위해 사용

conn 계정명/비밀번호 계정을 변경할 때 사용

\*conn을 진행 중에 실패한 경우 다시 로그인을 해야 한다. user는 ""입니다를 확인 가능

원하는 위치에 생성한 파일을 저장하기

- 명령프롬프트 바로가기 만들기

- 속성 바로가기 탭에서 시작위치를 원하는 폴더 위치로 지정한다.

- 바로가기를 실행하면 지정한 폴더 위치에서 시작할 수 있음

데이터베이스 가동:

sqlplus sys/change\_on\_install as sysdba

startup

데이터베이스 종료:

shutdown

데이터베이스 종료 중 무한 대기상태에 빠지게 되면 shutdown abort로 취소할 수 있음

존재하는 DB를 확인하는 순서

1. 어떤 테이블이 있는지 확인한다.

select \* from tab; <-- ; 을 반드시 입력해야 확인이 가능함

2. 컬럼구조를 확인한다. (dept 테이블을 확인할 경우)

desc dept; <-- ;을 쓰지 않아도 됨.

3. 컬럼명의 의미를 파악한다. (컬럼을 기준으로 데이터를 조회함으로 반드시 알아야 함)

데이터 형 (Type)

숫자

number 일반숫자

number(10) 10 자릿수까지의 숫자

number(8,2) 6자릿수의 숫자와 소숫점 이하 2자리까지 표시 (총 8자리수)

문자

char(10) 고정형 - 메모리를 고정된 형태로 차지함

varchar2(10) 가변형 - 필요없는 부분은 메모리에서 삭제한다. 낭비를 줄일 수 있음

날짜

Date 날짜 표시 (기본적으로 세기, 년, 월, 일, 시, 분, 초, 요일이 모두 저장되지만 일반적으로는 YY/MM/DD만 출력되며 필요할 때는 지

정하여 쓸수 있음)

데이터 조회

select 컬럼명 from 테이블명;

- 칼럼을 여러개 쓸 경우 구분자 ,를 쓴다.

- from 뒤에는 항상 ;을 붙인다.

- 전체를 보는 경우 \*로 대체 가능

한 행에 출력할 수 있는 문자 수가 80자로 제한되어 있는데 해당 수를 늘릴 수 있음,

set linesize 250

단 재시작하면 위 설정은 지워지므로 다시 입력해야 한다.

한 페이지에 출력할 수 있는 행 수가 10행으로 제한되어 있는데 해당 수를 늘릴 수 있음.

set pagesize 20

단 재시작하면 위 설정은 지워지므로 다시 입력해야 한다.

데이터 베이스에서 연산하기

산술연산자 + - \* / (\*나머지는 자바스크립트와 달리 %를 사용하지 않는다)

select에 연산자를 추가하여 출력하도록 할 수 있음

이때 연산자를 컬럼명으로 사용하여 출력된다.

NULL - 연산, 할당, 비교가 불가능한 값.

NULL을 nvl 함수를 이용하여 연산시에만 원하는 형식의 데이터로 치환해준다. (실제 데이터는 그대로 유지된다)

연산자를 컬럼명으로 사용하여 출력되는데 이를 as 별칭 <-- 을 추가하여 별칭을 부여할 수 있다.

원형 <-- select ename, sal, comm, sal\*12+comm, nvl(comm,0), sal\*12+nvl(comm,0) from emp;

별칭 <-- select ename, sal, comm, sal\*12+comm, nvl(comm,0), sal\*12+nvl(comm,0) "연봉" from emp;

또는 select ename, sal, comm, sal\*12+comm, nvl(comm,0), sal\*12+nvl(comm,0) as annsal from emp;

as는 생략가능함.

연산없이 문자열을 추가할 수 있음

|| <-- comcationation

|| '문장' || <-- '' 표시가 반드시 있어야지 실행이 가능

중복값을 제외하고 출력할 때 사용

Distinct

-----------------------------------------------------------160803-------------------------------------------------------

SQLplus에서 자주 사용하는 명령어

- 편집명령

LIST 이전에 실행한 명령문을 buffer에 저장했다가 보여줌. 단축형으로 L을 사용

/ 이전에 실행함 명령문을 바로 실행함

RUN L과 /를 동시에 실행 단축형으로 R을 사용

@

- 파일명령

EDIT 단축형으로 ED를 사용. 그냥 ED만 사용 시 버퍼에 저장된 내용이 메모장 팝업창으로 출력하며 쿼리문을 수정하고 메모창을 저장하

면서 닫으면 버퍼에 내용이 수정된다. 단, 가장 하단에 /를 삭제하면 안된다. 또한 파일명을 입력하면 버퍼와 마찬가지 방법으로 저장한

내용을 수정할 수 있으며 확장자 없이 입력하는 경우 /sql인 문서를 자동으로 확인한다. (가장 중요함)

HOST 버퍼를 유지한 상태에서 DOS 모드로 이동이 가능. EXIT로 다시 SQL로 돌아올 수 있음

SAVE 쿼리문을 파일형태로 저장했다가 향후에 사용할 수 있음. .sql 확장자를 사용하며 해당 확장자는 자동으로 생성됨으로 sql로 저장

하는 경우에는 확장자를 입력하지 않아도 된다. 파일생성은 처음 실행한 위치에 저장된다.

옵션으로 REPLACE (파일 덮어쓰기) 와 APPEND (이미 존재한 파일 쿼리문을 추가)가 있음

@ sql 확장자 파일에 저장된 쿼리문을 실행함. 확장자 생략할 경우 자동으로 .sql 파일을 찾음

GET 파일에 저장한 쿼리문을 버퍼로 가지고 옴. 확장자 생략할 경우 자동으로 .sql 파일을 찾음

SPOOL

- 데이터 베이스 접속 및 종료

HOST

EXIT

- 출력형식

SET 환경변수로 HEADING on/off, linesize 숫자, pagesize 숫자 등을 이용하여 출력형식을 변형한다.

COLUMN FORMAT은 컬럼값의 최대크기를 바꾸는 역활을 함. 형식은 column 칼럼명 format 변경을 원하는 형식

문자형식 지정할 경우: A를 사용하며 이하 숫자로 자릿수를 지정

숫자형식 지정할 경우: 9,999,999 또는 0,000,000을 사용한다. 9는 공란으로 0은 0으로 남는 자리를 채운다.

\*tip: 코드가 길어질수록 절 단위로 끊어서 사용한다.

Select로 측정 데이터 추출하기

select 컬럼명

from 테이블명

where 컬럼명 연산자 비교값

예) emp 테이블에서 급여가 3000 이상인 사람을 표시할 때

select \*

from emp

where sal >= 3000;

연산자

= 같다

> 크다

<작다

>= 크거나 같다

<= 작거나 같다

!= 다르다 (<>, ^= 도 사용하지만 주로 !=를 사용함)

값으로 숫자외 문자와 날짜를 사용할 수 있다.

문자는 '' 안에 입력해야 하며 대소문자를 가린다.

''를 입력하지 않은 경우에는 에러가 발생하며, 대소문자가 다른 경우에는 동일단어라도 다른 단어로 인식한다.

날짜 데이터도 ''를 사용해야 하며 날짜는 날짜 표시하는 형식에 맞춰서 지정해야 한다.

논리 연산자 <-- 연산자를 논리적인 관계로 연결한다.

And 두 조건을 모두 만족하는 경우만 검색하여 출력

Or 두 조건 중 한개만 만족하는 경우를 검색하여 출력

Not 조건에 만족하지 못하는 경우만 검색하여 출력하며 위 두 연산자와 다르게 컬럼 앞에 사용한다.

숫자가 조건인 경우에는 범위를 표현할 수 있음

컬럼명 between 값 and 값 으로 and 연산자를 표현할 수도 있음

or 연산자의 경우 컬럼명 not between 값 and 값 으로 표현할 수 있다. 값에 문자나 날짜가 오는 경우 마찬가지로 ''를 사용한다.

In 연산자

or 연사자를 간략하게 사용할 때 사용하는 방법

컬럼명 in(값1,값2,값3) = 컬럼명=값1 or 컬럼명=값2 or 컬럼명=값3

컬럼명 not in(값1,값2,값3) = 컬럼명!=값1 and 컬럼명!=값2 and 컬럼명!=값3

Like 연산자

와일드 카드를 값에 포함한다.

% 문자가 없거나 하나 이상의 문자에 어떤 값이 와도 상관없음

\_ 하나의 문자에 어떤 값이 와도 상관없음

escape 옵션 이하 한자리를 문자로 인식하게 함

order by 컬럼명 asc또는desc 기본은 asc로 생략이 가능

여러가지 정렬을 한번에 처리할 때는 ,로 구분자를 사용한다.

또한 order by는 반드시 마지막에만 사용이 가능하다.

-----------------------------------------------------------160804-------------------------------------------------------

dual 테이블 : 테스트 용 dummy 테이블

숫자함수

abs 절대값을 구한다

floor 소수점 아래를 버린다

round 지정한 자릿수 까지 표시하고 나머지는 반올림한다 (-로 표시하면 양수쪽으로 이동한다)

trunc round와 동일하게 진행되며 나머지는 버린다.

mod 나머지 구하기

문자처리 함수

upper 대문자로 변환

lower 소문자로 변환

initcap 이니셜만 대문자

length 순수하게 길이를 구할 때 사용

lengthb 바이트 수를 구할 때 사용 (한글은 2바이트를 사용, 영어는 1바이트를 사용)

substr (대상, 시작위치, 추출할 개수) 를 지정하여 문자를 추출한다. 시작위치가 -인 경우 뒤에서부터로 지정된다.

substrb 영문에 경우 substr과 동일하지만 한글은 2바이트를 사용하기 때문에 다른 결과가 발생한다.

instr 지정한 문자의 위치값을 구할 때 사용 기본은 첫번때 나오는 것을 표시 (instr(대상, 찾을 글자, 시작위치, 몇번째발견) 으로

다른 위치에 있는 동일한 글자도 표시 가능

instrb로 동일한 기능을 바이트를 가지고 진행

특정기호로 채우는 함수

LPAD 왼쪽에 여백을 주고 지정한 기호를 출력

RPAD 오른쪽에 여백을 주고 지정한 기호를 출력

LTRIM 지정한 곳의 왼쪽에 있는 여백을 제거

RTRIM 지정한 곳의 오른쪽에 있는 여백을 제거

TRIM 지정한 곳의 양쪽에 있는 여백을 제거, 또는 지정한 문자를 제거

날짜 함수

sysdate 시스템에 저장된 현재날짜를 반환하는 기능으로 괄호 없이 사용한다. 결과값은 일자로 구분된다.

round(날짜, 형식) 형식의 기준으로 반올림

cc, scc 4자리 연도의 끝의 2자리를 기준으로 반올림

syyy,yyyy,year, syear, yyy,yy,y 년 기준 반올림

ddd,d,j 일을 기준

hh,hh12,hh24 시를 기준

q 한 분기의 두번 째 달의 16일을 기준으로 반올림

month, mon, mm, rm 월 16일을 기준으로 반올림

day, dy, d 한주가 시작되는 날짜

mi 분을 기준

trunc(날짜, 형식) 형식의 기준으로 버림. 형식은 round와 동일

last\_day 해당 월의 마지막날짜를 출력

형변환 함수

TO\_CHAR (날짜데이터, '출력형식')

날짜를 문자 형태로 표현

\* 출력형식

yyyy 4자리 연도

yy 2자리 연도

mm 숫자 월표현

mon 월표시 (알파벳표현)

day 요일 표현

dy 요일 축약 표현

hh 또는 hh12 12시간으로 표시

am 또는 pm 오전 오후 표시

a.m 또는 p.m 오전 오후 표시

hh24 24시간 으로 표시

mi 분으로 표시

ss 초로 표시

숫자를 문자 형태로 표현 <-- 연산이 불가능함

\* L 통화 기호 표시

, 천 단위 구분

. 소수점

TO\_DATE ('문자', '형식')

데이터와 검색하고자 하는 기준값의 자료형태를 맞춰야지만 검색이 가능하다.

하여 주로 날짜 데이터를 검색이나 연산해야 할 때 사용

TO\_NUMBER

숫자형태로 변경하여 표현

nvl 함수

nvl(컬럼명, 변경값) 컬럼명과 변경값은 형식이 같아야 한다.

만약 변경값은 문자이나 컬럼명의 형식이 숫자인 경우에는 컬럼명의 형식을 to\_char를 이용하여 문자로 변경하여 출력한다.

-----------------------------------------------------------160805-------------------------------------------------------

조건식이 포함된 함수들

Decode 함수

decode (표현식, 조건1, 결과1, 조건2, 결과 2, 조건3, 결과 3, 기본결과n)

표현식과 조건값을 비교연산(=)을 한다. 모든 조건이 만족하지 않는 경우 기본결과n이 출력되며 기본결과n은 생략가능.

Case 함수

case when 조건1 then 결과1

when 조건2 then 결과2

when 조건3 then 결과3

else 결과n

end

단순비교하는 decode와 다르게 범위비교가 가능한 장점이 있으며 if else와 유사한 구조를 가짐

-----------------------------------------------------------160808-------------------------------------------------------

그룹함수

SUM (컬럼)

여러개의 데이터의 합계를 구함

Null 값은 자동으로 제외하고 연산한다.

AVG (컬럼)

평균값을 구한다.

Null 값은 자동으로 제외하고 연산한다.

Max (컬럼)

최대값을 구한다.

Null 값은 자동으로 제외하고 연산한다.

Min (컬럼)

최소값을 구한다.

Null 값은 자동으로 제외하고 연산한다.

count (컬럼)

null 데이터를 제외하고 연산한다.

count(\*)를 이용하여 전체 로우 개수를 출력

GROUP BY

select 컬럼명, 또는 그룹함수

from 테이블명

where 조건 (연산자)

group by 컬렴명;

group by 절에 명시된 컬럼은 개수가 다르지 않기 때문에 그룹함수와 같이 select절에 사용이 가능하다. 만약 개수가 같지 않는 경우에

는 역시 사용이 불가능하다.

Having 조건

단독으로는 사용이 불가능하며 group by와 같이 사용한다.

group by 로 구분할 때 구분되는 컬럼에 대한 특정조건을 추가로 명시할 때 사용한다.

문제

- HR 계정으로 접속

- employees 테이블에서 확인

- 부서별 근부인원 수를 구한다

- 단 부서인원이 5명 이하인 인원만 구한다.

- 부서번호로 정렬하고 Null값은 제외한다.

풀이

select department\_id, count(department\_id) 근무자수

from employees

where department\_id is not null

group by department\_id

having count(department\_id) <=5

order by department\_id asc

/

\*order by 절은 항상 마지막에만 사용이 가능하다.

JOIN

테이블블 같이 보기 위해서는 from 절에 ,로 추가 한다.

- cross join

전체를 보는 방식으로 a테이블 X b테이블의 데이터 수를 보여준다

잘 사용하지 않는다.

- equi join

where 조건절이 반드시 들어간다

공통되는 컬럼을 명기한다.

where a.공통컬럼 = b.공통컬럼;

-----------------------------------------------------------160809-------------------------------------------------------

- equi join

where 조건절이 반드시 들어간다

공통되는 컬럼을 명기한다.

where a테이블.공통컬럼 = b테이블.공통컬럼;

테이블에 별칭을 사용하여 소속테이블을 지정할 수 있음

- non-equi join

범위조건으로 비교하기 위해서 사용

3가지 테이블을 동시에 연결하기도 함

예제)

select e.ename, d.dname, s.grade

from emp e, dept d, salgrade s

where e.deptno = d.deptno

and e.sal between s. losal and s.hisal

/

- self join

마치 2개의 테이블이 있는 것처럼 논리적으로 분리시킨 후 join하는 것처럼 사용

테이블이 실제로는 1개 이므로 반드시 별칭을 사용한다.

from 테이블명 별칭a, 테이블명 별칭b

- outer join

NULL값 또는 한 테이블에만 있는 데이터로 인하여 데이터가 누락되는 문제를 해결하기 위하여 사용

(+)에서 정보가 부족한 컬럼쪽에 붙여준다.

예를 들면 별칭a.컬럼 = 별칭b.컬럼 에서 별칭a의 데이터가 별칭b에서 찾을 수 없는 경우 self join에서는 데이터가 누락되어 표시

되는데, 이때 outer join을 사용하여 별칭a.컬럼 = 별칭b.컬럼 (+) 로 표시하면 전체 데이터를 모두 표시할 수 있다.

ANSI JOIN

- join관련의 새로운 표준방식

, 를 대신하여 명령어를 넣는다.

cross join

select \*

from emp cross join fept;

inner join

select ename, dname

from emp inner join dept

on emp.deptno-dept.deptno;

- inner join의 조건 지정

select ename, dname

from emp inner join dept

on emp.deptno=dept.deptno

where ename='SCOTT'

- on emp.deptno=dept.deptno 같이 컬럼명이 같을 경우 using을 이용하여 간단하게 표기 할 수 있음

using (deptno)

- natural join의 조건 지정

비교하려는 테이블에서 1개의 칼럼만이 공통적으로 같은 이름을 사용할때만 사용한다.

테이블을 그런 의도로 만든 경우에만 주로 사용한다.

-----------------------------------------------------------160810-------------------------------------------------------

- Outer join

from 테이블A [left / right / full ] outer join 테이블B

full의 경우 양쪽 테이블에서 모두 단독 데이터가 있을 경우 사용이 가능하다.

일반 outer join의 경우 양쪽 모두에 (+) 지정할 수 없다.

문제

1. HR 계정으로 접속

2. 아래 컬럼을 출력

FIRST\_NAME, LAST\_NAME, EMAIL, DEPARTMENT\_ID, DEPARTMENT\_NAME, JOB\_ID, JOB\_TITLE, CITY

3. 107명이 전부 나오게 한다

4. 일반 OUTER JOIN을 사용한다.

5. 이용할 테이블의 별칭은 EMPLOYEES EMP, DEPARTMENTS DEP, JOBS JOB, LOCATIONS LOC 를 사용한다.

풀이1 - 일반 Outer Join

select emp.FIRST\_NAME, emp.LAST\_NAME, emp.EMAIL, emp.JOB\_ID, job.job\_title,dep.department\_ID, dep.department\_name, loc.city

from employees emp, jobs job, departments dep, locations loc

where emp.job\_id=job.job\_id

and emp.department\_id = dep.department\_id (+)

and dep.location\_id = loc.location\_id (+)

풀이2 - ANSI JOIN Outer Join

select emp.FIRST\_NAME, emp.LAST\_NAME, emp.EMAIL, emp.JOB\_ID, job\_title,dep.department\_ID, dep.department\_name, loc.city

from employees emp left outer join jobs job on emp.job\_id = job.job\_id

left outer join departments dep on emp.department\_id = dep.department\_id

left outer join locations loc on dep.location\_id = loc.location\_id

-----------------------------------------------------------160811-------------------------------------------------------

서브 쿼리

단일행 서브쿼리

단일 값을 비교하는 연산자를 사용하는 것 =, >, <. >=, <= <> 등을 사용

메인쿼리 중 where 절의 컬럼값에 또다른 쿼리를 삽입하는 방식

select 컬럼

from 테이블

where 조건 = (select 컬럼 from 테이블 where 조건 = 조건값)

문제1

1. 월급이 가장 적은 사원 조회

2. first\_name, last\_name, job\_title 컬럼 출력

3. 조인과 서브쿼리 사용

풀이

select first\_name, last\_name, job\_title

from employees inner join jobs

using (job\_id)

where salary=(select min(salary) from employees)

문제2

1. 근무지가 US인 사원의 급여평균보다 많이 받는 사원 조회

Employees: first\_name, last\_name, department\_ID,

Locations: location\_ID, country\_ID

Departments: department\_ID, Location\_ID

풀이

근무지가 US인 사원

select first\_name, last\_name, country\_ID, salary

from employees left outer join departments

using (department\_id)

left outer join locations

using (location\_id)

where country\_id='US'

US 급여평균

select avg(salary)

from employees left outer join departments

using (department\_id)

left outer join locations

using (location\_id)

where country\_id='US'

최종

select first\_name, last\_name, job\_title

from employees inner join jobs

using (job\_id)

where salary>(select avg(salary)

from employees left outer join departments

using (department\_id)

left outer join locations

using (location\_id)

where country\_id='US')

다중행 서브쿼리

아래 연산자를 사용

in 다양한 값 중에서 한가지라도 참이면 결과를 구함

all 연산기호를 반드시 사용해야 한다. 부재일 경우 에러출력됨

-----------------------------------------------------------160812-------------------------------------------------------

any 메인 쿼리에 조건 중 한 개라도 일치하면 참으로 처리. 연산기호를 반드시 사용해야 한다.

some

exist

영업사원들의 최소 급여

SALESMAN

ename, sal, job

DDL 구문 : 테이블 구조 생성, 변경, 삭제에 사용

Create 객체를 생성한다. 테이블도 객체에 속함.

형식 - Create table 테이블명 (컬럼명1, 데이터종류1, 컬럼명2, 데이터종류2, 등등)

최종으로 사용하는 컬럼 이후 데이터 종류 되에는 ,가 오지 앖는다.

tiemestamp(n) date의 확장된 형테

식별자 명명 규칙

반드시 문자로만 시작 가능 (영문 대소문자)

숫자와 \_ $ # 만 허용

공백허용안함

서브쿼리로 테이블을 생성하는 방법

기존 테이블의 전체나 일부를 가지고 와서 생성하는 방법

형식 - create table 테이블명 as select구문

이 경우 데이터도 함께 가지고 온다.

테이블의 구조만 복사하기

형식 - create table 테이블명 as select구문 where 1=0;

ALter

테이블 구조를 변경한다.

컬럼을 변경, 삭제, 추가한다.

형식 -

컬럼추가: alter table 테이블명 add (컬럼명, 데이터타입)

속성변경: alter table 테이블명 modify (컬럼명, 데이터타입)

\* 컬럼에 데이터가 없는 경우 변경이 가능하지만 데이터가 있는 경우에는 크기만 늘릴 수 있고 줄일 수는 없다.

컬럼삭제: alter table 테이블명 drop column 컬럼명

컬럼명변경: alter table 테이블명 rename column 기존컬럼명 to 변경컬럼명

Drop

drop table 테이블명;

테이블을 삭제한다.

삭제할 경우 윈도우의 휴지통과 같은 역활을 하는 곳으로 이동된다.

table을 검색할 경우 정보확인이 가능하다.

이 기능을 활용할 수 있는 추가 명령어

1. 휴지통에서 복원하기

flashback table 테이블명

to before drop;

2. 휴지통 비우기

purge recyclebin;

3. 휴지통 내용 확인하기

show recyclebin;

truncate

테이블은 그대로 두고 데이터만 삭제한다.

형식 - truncate table 테이블명

rename

테이블의 이름이 변경되는 것

형식 - rename 기존테이블명 to 변경할테이블명

데이터 딕셔너리와 데이터 딕셔너리 뷰

접두어 (권한에 따라서 분류) (XXXX는 객체의 종류)

DBA\_XXXX DBA 계정으로만 확인이 가능

ALL\_XXXX 타 계정의 객체에 대하여 위임을 받은 정보 조회를 할 수 있음

USER\_XXXX 내가 소유한 객체에 대한 정보 조회

select \* from tab; 약어로 정리된 것.

원래는 select table\_name from user\_tables 로 사용해야 하는 것임

테이블의 내용을 추가, 수정, 삭제하는 DML

데이터 삽입

- 형식: insert into 테이블명 (컬럼명,~) values (컬럼값,~)

- 값의 개수와 컬럼명의 개수와 타입이 일치해야 함.

- 문자를 사용할 때는 '' 를 사용한다.

- 컬럼 전체를 다 사용하는 경우 생략이 가능

-

-----------------------------------------------------------160816-------------------------------------------------------

Null 값을 넣기

- 암시적으로 넣기

null 값이 들어갈 컬럼을 제외한 나머지 컬럼과 값에만 지정

- 명시적으로 넣기

NULL 이라고 적거나 '' 을 적는다. ''와 ' '은 다름.

서브쿼리로 삽입하기

insert into 테이블명 select 구문

검색되는 select문의 결과 값이 입력된다.

다중 테이블에 다중행 입력

insert all

into 테이블1 (컬럼명a,컬럼명b,컬럼명c)

into 테이블2 (컬럼명a,컬럼명b,컬럼명d)

select 컬럼명a, 컬럼명b, 컬럼명c, 컬럼명d

from 테이블

where 조건

update 구문

update 테이블명

set 컬럼= 값

where 조건절

\* 조건절이 없는 경우 해당 컬럼의 모든 데이터를 변경한다.

\* 조건절이 있는 경우 해당 조건에 부합하는 컬럼의 데이터만을 변경한다.

2개 이상의 컬럼값 변경

update 테이블명

set 컬럼1= 값1, 컬럼2=값2...

where 조건절

\* 여러 값을 한번에 변경도 가능하며 이때 기존 값과의 동일유무는 상관 없이 입력한 값으로 입력된다.

서브쿼리는 아래 형식으로 이용가능

update 테이블명

set 컬럼1= (select구문1), 컬럼2= (select구문2),,,,,

where 조건절

혹은

update 테이블명

set (컬럼1,컬럼2, 컬럼3,,,,)= (select구문)

where 조건절

혹은

set 컬럼= 값1

where 조건절(select구문)

문제1

1. hr 계정으로 접속

2. 아래 방법으로 테이블을 생성

CREATE TABLE month\_salary (

magam\_date DATE not null,

department\_id number,

emp\_count number,

total\_salary number(9,2),

average\_salary number (9,2)

);

3. employees 컬럼을 이용해서 이번달 말일을 입력한다. 서브쿼리를 이용

4. employees 컬럼에서 department\_Id 그룹별로 한개씩 넣는다. group by를 이용

풀이1

insert into month\_salary (magam\_date, department\_id)

select last\_day(sysdate), department\_id from employees group by department\_id

문제2

1. 문제1 이후 부서번호 별 사원수, 총액, 평균을 Employees 테이블에서 서브쿼리문을 입력한다.

update month\_salary a

set a.emp\_count = (select count(\*) from employees k where k.department\_id = a.department\_id

group by k.department\_id),

a.total\_salary = (select sum(k.salary) from employees k where k.department\_id = a.department\_id

group by k.department\_id),

a.average\_salary = (select round(avg(k.salary)) from employees k where k.department\_id = a.department\_id

group by k.department\_id)

update month\_salary a

set (a.emp\_count,a.total\_salary,a.average\_salary) = (select count(\*), sum(k.salary),round(avg(k.salary)) from employees k

where k.department\_id = a.department\_id

group by k.department\_id)

-----------------------------------------------------------160817-------------------------------------------------------

delete 구문

형식

delete from 테이블명

where 조건절

조건절에는 서브쿼리로 대체할 수 있음

트랜잭션

데이터를 처리하는 일련의 작업단위

select, insert, update, delete 같은 명령어의 사용에 이용되는 것

실제 데이터베이스에 반영시키는 방법

commit

- 데이터를 정상적으로 처리하겠다고 확정하는 명령어

- 데이터를 확정시키는 역활

savepoint 중간에 포인트를 두어 중간과정부터 다시 하도록 하는 방법

rollback 마지막 commit 또는 rollback 실행한 지점에서 처음부터 다시 하는 방법

Create alter drop 등 DDL, DCL 명령문 후에는 자동으로 commit이 실행되게 된다.

비정상 종료된 경우 자동으로 rollback 된다.

여러 사용자가 한 개의 데이터 베이스를 이용하는 경우

작업한 내용이 위에 자동으로 commit이 되는 경우를 제외하고는 commit 또는 rollback등을 통하여

작업내용을 공유해야만 공유가 가능해 진다.

trancate와 delete의 차이점을 확인할 것

데이터 무결성을 위한 제약 조건

데이터 자체에는 문제가 없지만 무결한 데이터로만 유지할 수 있도록 사전에 차단한다.

\*테이블의 제약조건 분석\*

column constraint\_type format A18 <--컬럼의 크기를 넓혀주는 역활

select constraint\_name, constraint\_type, table\_name

from user\_constraints

column owner format a10

column table\_name format a15

column column\_name format a15

select \*

from user\_cons\_columns;

NOT NULL

Null 데이터를 사용하지 않는다.

약어 : C

지정방법: 테이블 생성 또는 변경할 때 아래 같은 방법으로 진행한다.

컬럼명 데이터타입 not null

unique

중복 값은 사용이 불가능 하다.

약어 U

지정방법 : 컬럼명 데이터타입 unique

unique만으로는 null을 데이터로 인정하며 중복적으로 사용된다.

만약 null도 사용하지 못하게 하려면 두가지를 모두 병기 하거나 primary key를 사용한다.

컬럼명 데이터타입 unique not null

또는 컬럼명 데이터타입 primary key

----------------------------------------------------160818-------------------------------------------------

**제약 조건명 설정하기**

**CONSTRAINT**

권장형식: 테이블명\_컬럼명\_제약조건유형

이름은 개발자가 임의지정한다.

제약조건을 지정하면 오류메시지에 제약 조건명이 그대로 표시되어 원인을 쉽게 확인할 수 있음.

제약 조건을 지정할때 제약 조건명을 추가하여 설정한다.

형식: create table 테이블명 (컬럼명 형식 constraint 제약조건명 제약조건)

**PRIMARY KEY**

기본 키로 not null+unique 의 제약조건을 모두 사용한다. 테이블에서 한개 이상의 컬럼에는 반드시 사용하는 것이 일반적이다.

***약어 P***

**FOREIGN KEY**

2개 이상의 테이블들의 관계 (Entity Relation Diagram)를 이용해서 제약을 설정한다.

부모키와 자식키가 중요한 개념이며 관계를 맺는 각 테이블마다 공통컬럼이 있음

데이터를 삽입할 때 필요한 정보를 최소화할 수 있음

데이터 타입을 맞춰주어야 하며 최소 부모키와 자식키의 타입크기는 같거나 자식키가 커야 함.

제약조건은 한쪽 테이블에만 지정하는데 이때 자식키 쪽에서 지정한다.

N:1 구조 , N:M 구조 등으로 만들 수 있다.

***약어 R***

지정방법 : 자식컬럼 references 부모테이블(부모컬럼)

\*부모테이블의 참조할 컬럼은 반드시 Primary Key로 지정해야 하고 자식컬럼은 부모 테이블의 데이터만을 참조하고 특성은 따르지는 않는다. 하여 NULL 값도 입력이 가능하다. NULL 값을 입력할 수 없도록 하기 위해서는 not null을 지정한다.

**과제:** 관계 다이어그램 10가지 찾아서 PPT로 그리기

|  |
| --- |
| select table\_name, constraint\_type, constraint\_name, r\_constraint\_name  from user\_constraints  where table\_name in ('DEPT', 'EMP') |

<EMP와 DEPT 테이블의 제약조건 확인>

**CHECK**

데이터 범위를 설정한다. (M 또는 F 선택, 1~100 사이의 숫자만 사용 등)

***약어 C***

SEARCH CONDITION으로 제약조건을 확인할 수 있다.

|  |
| --- |
| select table\_name, constraint\_type, constraint\_name, search\_condition  from user\_constraints  where table\_name='EMP07' |

<EMP07의 CHECK 제약조건 확인>

**DEFAULT 제약조건 설정하기**

특정 컬럼에 null 값이 입력될 때 자동으로 지정한 값으로 치환하여 입력되도록 설정한다.

형식: 컬럼 타입 default ‘지정값’

|  |
| --- |
| select table\_name, column\_name, data\_default  from user\_tab\_columns  where table\_name='DEPT01' |

<DEPT01의 default 제약조건 확인>

**테이블 레벨 방식으로 제약조건 지정하기**

형식:

Create Table 테이블 명 (

컬럼명1 데이터타입1,

컬럼명2 데이터타입2,

컬럼명3 데이터타입3,

컬럼명4 데이터타입4,

constraint 제약조건명 제약조건(컬럼명1),

제약조건(컬럼명2)

)

\* 제약조건명은 생략 가능

사용하는 경우:

1. Primary Key를 1개 이상 설정하려고 하는 경우
2. Alter table로 제약 조건을 추가할 경우

\*FOREIGN KEY의 경우 FOREIGN KEY를 그대로 명령어로 사용한 후 이하 references를 기입한다.

형식: foreign key (자식컬럼명) references 부모테이블(부모컬럼명)

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name,R\_constraint\_name  from user\_constraints  where table\_name='EMP03' |

<EMP03의 제약조건 확인>

Primary Key를 2개 이상 지정하기 위해서는 아래 형식으로 지정한다.

형식: constraint 제약조건명 Primary key (컬럼1, 컬럼2)

**제약조건 변경**

**alter테이블에 add절을 이용한다.**

형식:

*Alter\_table 테이블명*

*Add constraint 제약조건명 제약조건(컬럼명)*

**rename절을 이용하여 제약조건의 이름을 변경할 수 있음**

형식:

*alter table 테이블명*

*rename constraint 기존제약조건명 to 새로운제약조건명*

**NOT NULL을 제약조건으로 추가하기**

테이블을 처음 생성할 때 NULL 값을 사용하는 것을 기본적으로 지정되어 생성이 되기 때문에 add절을 이용할 수가 없다. 하여 modify를 이용하여 변경해 주어야 한다.

형식:

*alter table 테이블명*

*modify 컬럼명 constraint 제약조건명 not null*

**제약조건을 제거하기**

형식:

*alter table 테이블명*

*drop constraint 제약조건명*

\*제약조건명을 모르더라도 PRIMARY KEY를 이용하여 삭제가 가능하다.

**제약조건 활성/비활성**

자식테이블에서 현재 데이터를 참조하고 있는 경우 부모테이블의 데이터를 임의로 제거할 수가 없다

----------------------------------------------------160819-------------------------------------------------

**비활성화 시키기**

형식:

*ALTER TABLE 테이블명*

*DISABLE constraint 제약조건명*

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name, R\_constraint\_name, Status  from user\_Constraints  where table\_name='EMP01' |

<EMP01의 활성화 여부 확인>

**활성화 시키기**

형식:

*ALTER TABLE 테이블명*

*ENABLE constraint 제약조건명*

비활성화한 컬럼을 활성화할 때는 자식컬럼에 데이터 유무를 확인해야 하며, 데이터가 존재하는경우 부모의 데이터를 추가하거나 자식 컬럼에 해당 데이터를 제외하는 방법으로 무결성을 완성해야 가능하다.

**CASCADE**

부모자식간의 종속관계가 유지된 테이블에서 foreign key와 primary key를 모두 제거할 때 쓰는 옵션

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name, R\_constraint\_name, status  from user\_constraints  where table\_name in ('DEPT01', 'EMP01') |

<EMP01과 DEPT01의 제약조건들에 대한 활성화 여부 확인>

문제

1. HR 계정으로 접속
2. EMPLOYEES 테이블과 똑같은 구성으로 EMPTEST 테이블 만들기
3. 테이블 레벨방식으로 제작 (제약조건 포함)

풀이

1. 데이터 타입확인

|  |
| --- |
| DESC EMPLOYEES |

|  |  |  |
| --- | --- | --- |
| 이름 | Null | 유형 |
| EMPLOYEE\_ID | NOT NULL | NUMBER(6) |
| FIRST\_NAME |  | VARCHAR2(20) |
| LAST\_NAME | NOT NULL | VARCHAR2(25) |
| EMAIL | NOT NULL | VARCHAR2(25) |
| PHONE\_NUMBER |  | VARCHAR2(20) |
| HIRE\_DATE | NOT NULL | DATE |
| JOB\_ID | NOT NULL | VARCHAR2(10) |
| SALARY |  | NUMBER(8,2) |
| COMMISION\_PCT |  | NUMBER(2,2) |
| MANAGER\_ID |  | NUMBER(6) |
| DEPARTMENT\_ID |  | NUMBER(4) |

1. 제약조건명 및 조건 확인

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name  from user\_constraints  where table\_name = ‘EMPLOYEES’ |

|  |  |  |
| --- | --- | --- |
| CONSTRAINT\_NAME | CONSTRAINT\_TYPE | TABLE\_NAME |
| EMP\_MANAGER\_FK | R | EMPLOYEES |
| EMP\_JOB\_FK | R | EMPLOYEES |
| EMP\_DEPT\_FK | R | EMPLOYEES |
| EMP\_LAST\_NAME\_NN | C | EMPLOYEES |
| EMP\_EMAIL\_NN | C | EMPLOYEES |
| EMP\_HIRE\_DATE\_NN | C | EMPLOYEES |
| EMP\_JOB\_NN | C | EMPLOYEES |
| EMP\_SALARY\_MIN | C | EMPLOYEES |
| EMP\_EMAIL\_UK | U | EMPLOYEES |
| EMP\_EMP\_ID\_PK | P | EMPLOYEES |

1. 각 제약조건이 지정된 컬럼명 확인

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name  from user\_constraints  where table\_name = ‘EMPLOYEES’ |

|  |  |  |  |
| --- | --- | --- | --- |
| CONSTRAINT\_NAME | CONSTRAINT\_TYPE | TABLE\_NAME | COLUMN\_NAME |
| EMP\_MANAGER\_FK | R | EMPLOYEES | MANAGER\_ID |
| EMP\_JOB\_FK | R | EMPLOYEES | JOB\_ID |
| EMP\_DEPT\_FK | R | EMPLOYEES | DEPARTMENT\_ID |
| EMP\_LAST\_NAME\_NN | C | EMPLOYEES | LAST\_NAME |
| EMP\_EMAIL\_NN | C | EMPLOYEES | EMAIL |
| EMP\_HIRE\_DATE\_NN | C | EMPLOYEES | HIRE\_DATE |
| EMP\_JOB\_NN | C | EMPLOYEES | JOB\_ID |
| EMP\_SALARY\_MIN | C | EMPLOYEES | SALARY |
| EMP\_EMAIL\_UK | U | EMPLOYEES | EMAIL |
| EMP\_EMP\_ID\_PK | P | EMPLOYEES | EMPLOYEE\_ID |

1. CHECK 제약 확인

|  |
| --- |
| select constraint\_name, constraint\_type, table\_name  from user\_constraints  where table\_name = ‘EMPLOYEES’ |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CONSTRAINT\_NAME | CONSTRAINT\_TYPE | TABLE\_NAME | COLUMN\_NAME | SERCHCONDITION |
| EMP\_MANAGER\_FK | R | EMPLOYEES | MANAGER\_ID |  |
| EMP\_JOB\_FK | R | EMPLOYEES | JOB\_ID |  |
| EMP\_DEPT\_FK | R | EMPLOYEES | DEPARTMENT\_ID |  |
| EMP\_LAST\_NAME\_NN | C | EMPLOYEES | LAST\_NAME | “LAST\_NAME” IS NOT NULL |
| EMP\_EMAIL\_NN | C | EMPLOYEES | EMAIL | “EMAIL” IS NOT NULL |
| EMP\_HIRE\_DATE\_NN | C | EMPLOYEES | HIRE\_DATE | “HIRE\_DATE” IS NOT NULL |
| EMP\_JOB\_NN | C | EMPLOYEES | JOB\_ID | “JOB\_ID” IS NOT NULL |
| EMP\_SALARY\_MIN | C | EMPLOYEES | SALARY | SALARY > 0 |
| EMP\_EMAIL\_UK | U | EMPLOYEES | EMAIL |  |
| EMP\_EMP\_ID\_PK | P | EMPLOYEES | EMPLOYEE\_ID |  |

CREATE TABLE EMPTEST

EMPLOYEE\_ID NUMBER(6)

FIRST\_NAME VARCHAR2(20)

LAST\_NAME VARCHAR2(25) NOT NULL

EMAIL VARCHAR2(25) NOT NULL

PHONE\_NUMBER VARCHAR2(20)

HIRE\_DATE DATE NOT NULL

JOB\_ID VARCHAR2(10) NOT NULL

SALARY NUMBER(8,2)

COMMISION\_PCT NUMBER(2,2)

MANAGER\_ID NUMBER(6)

DEPARTMENT\_ID NUMBER(4)

CONSTRAINT EMPT\_EMP\_ID\_PK PRIMARY KEY (EMPLOYEE\_ID)

CONSTRAINT EMPT\_EMAIL\_UK UNIQUE (EMAIL)

CONSTRAINT EMPT\_SALARY\_MIN CHECK(SALARY > 0)

CONSTRAINT EMPT\_JOBID\_NN CHECK(JOIBID IS NOT NULL)

CONSTRAINT EMPT\_HIRE\_DATE\_NN CHECK(HIRE\_DATE IS NOT NULL)

CONSTRAINT EMPT\_EMAIL\_NN CHECK(EMAIL IS NOT NULL)

CONSTRAINT EMPT\_LAST\_NAME\_NN CHECK(LAST\_NAME IS NOT NULL)

CONSTRAINT EMPT\_DEPT\_FK FORIEGN KEY (DEPARTMENT\_ID) REFERENCES

CONSTRAINT EMPT\_JOB\_FK FORIEGN KEY (JOB\_ID) REFERENCES

CONSTRAINT EMPT\_ MANAGER\_FK FORIEGN KEY (MANAGER\_ID) REFERENCES