**DB 공부 정리**

전채현

**쿼리문 정리 (기본 CRUD)**

C : Create -> 추가

R : Read -> 읽기, 검색

U : Update -> 수정

D : Delete -> 삭제

**시퀀스 작성 방법**

create sequence cafe\_seq

start with 1// 1부터 시작

maxvalue 999999999// 마지막 번호

increment by 1// 증가 값

nocache// 시퀀스를 빨리 제공하기 위해 메모리에 캐쉬 지정 (nocache 일 경우 20으로 고정)

nocycle;// 최대값 도달 시 시퀀스 생성 중단

**테이블 작성 방법**

create table cafe(

idx number default CAFE\_SEQ.nextval primary key,// 시퀀스를 사용할 때는 앞에 default를 붙여줘야 한다

name varchar2(1000) not null,

menu varchar2(1000) not null

);

**테이블에 값 추가**

1. 시퀀스를 넣어서 작성하는 방법

insert into [테이블이름] values (테이블에서 작성했던 컬럼 순서대로 값 넣어주기)

insert into cafe values (cafe\_seq.nextval, '이디야','아메리카노');

2. 시퀀스를 안넣고 지정한 컬럼값으로 작성하는 방법

insert into [테이블 이름(컬럼명, 컬럼명)] values (값1, 값2);

insert into cafe(name, menu) values ('스타벅스','돌체라떼');

**select 문 정리**

select \* from [테이블 명] where [컬럼 명] order by [정렬할 컬럼] desc/asc

**불러오기**

select \* from cafe;

테이블 값 불러오기 idx가 낮은 순서대로

select \* from cafe order by idx;

**검색하기**

select \* from [테이블 명] where [검색할 컬럼] = 검색할 값;

select \* from cafe where idx = 1;

**수정하기**

update [테이블 명] set [컬럼 명]='수정할 값', [컬럼 명]='수정할 값' where [수정할 idx] = 수정할 idx 값;

update cafe set name='커피빈', menu='카페라떼' where idx = 1;

**삭제하기**

delete from [테이블 명] where [컬럼 명] = 컬럼 값;

delete from cafe where idx = 1;

**주의할 점!!**

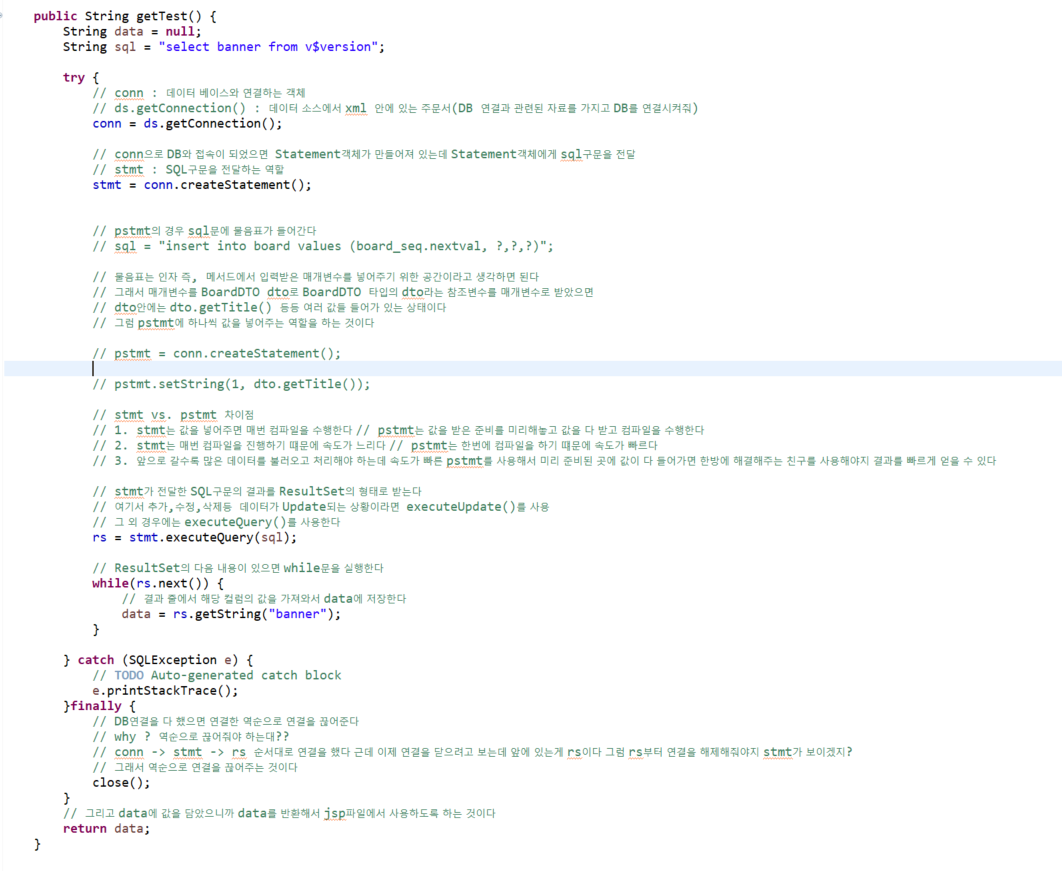
수정이나 삭제 시 primary key로 삭제나 수정하는 것이 좋다

why?? 이름이나 메뉴는 겹치는게 있을 수 있기 때문에 특정 1개의 값을 수정하거나 삭제하기 위해서는

그 값에 지정된 고유의 값으로 수정, 삭제해주는 것이 바람직 하다.

-> 아이디 삭제할 때 주민번호를 입력해야 삭제할 수 있는 느낌

===========================================



**그림 1 EclipseDB 연결하는 방법**

**Synoym**

Oracle에는 Synonym이란 객체가 있습니다. -> Synoym 시너님

객체에 대해서 영속적으로 별명을 지어주는 것.

테이블 ALIAS와는 다르게 임시적인 별명이 아닌 영속적으로 별명을 부여하는 것이 Synonym입니다.

Oracle Database를 사용해보면서 한번도 Synonym을 만들어보지 않아서 사용을 한번도 안했다! 라고 생각하실수도 있는데...

우리는 이미 Synonym을 사용하고 있습니다.

1

SELECT SYSDATE FROM DUAL;

DUAL이라는 임시 테이블은 SYNONYM입니다.

DUAL테이블은 모든 계정에서 스키마 없이 DUAL이라는 명칭으로 사용되고 있습니다.

DUAL도 임시적으로 사용되는 테이블인데 왜 스키마 없이 DUAL이라는 명칭으로 사용이 가능했을까요?

DUAL 테이블은 어느 계정에 소유된 테이블일까요?

1

SELECT SYSDATE FROM SYS.DUAL;

DUAL 테이블은 사실 SYS 계정이 소유하고 있는 테이블입니다.

Public 권한으로 동의어를 생성하여 모든 계정에서는 DUAL이라는 Synonym을 사용하는 것이었습니다.

===========================================

**Synonym 사용 이유**

1. 테이블 소유 보안

DUAL 테이블 처럼 긴이름을 짧게 혹은 짧은 이름을 길게 사용하기 위해 Synonym을 이용하기도 하지만,

실제로는 보안의 목적으로 많이 사용되고 있습니다.

위 이미지처럼 사용자가 EMP라는 테이블에 대한 정보가 필요할때 테이블이 어떤 스키마에 있는지 노출시켜서는 안됩니다.

그래서 EMP라는 SYNONYM을 생성하여 해당 SYNONYM에 대한 권한을 사용자에게 부여하여 보안을 강화할 수 있습니다

2. Database 위치 투명성

Database에 접근하기 위해서는 Protocol, IP, Listener Port, SID, 계정 정보를 들고 Database Link를 통해 결과를 받아옵니다.

DEPT라는 테이블은 USA에 있는 정보이고 EMP 테이블은 한국에 있는 정보라고 가정해보겠습니다.

SELECT \* FROM EMP;

현재 접속한 유저의 권한으로 EMP 테이블을 읽으면 LOCAL에 있는 Database에 Schema를 읽겠다는 것을 의미합니다.

그런데 USA에 있는 DEPT테이블을 읽어야하는 상황이라면

SELECT \* FROM DEPT@USA;

USA라는 DB LINK를 생성하여 @USA로 DB를 조회해야합니다.

사용자 측에서 @USA DB LINK 정보를 같이 사용하도록 데이터베이스에 대한 정보를 제공한다면 DEPT 테이블은 USA에 존재하고

EMP 테이블은 한국에 존재한다는 것을 알게 됩니다.

위 처럼 Database의 위치 투명성에 위반되는 문제를 해결하기 위해 Synonym을 생성해줄 때

DB Link에 대한 정보를 넣어 생성하고 Synonym에 대한 정보만 제공한다면 사용자로부터 Database의 위치를 투명하게 가려줄 수 있습니다.

3. 업무적 편의

업무적 편의를 위해 사용되기도 합니다.

HR SCHEMA에서 사원(EMP)에 대한 정보를 관리하고 있었습니다.

그런데 회사 정책상 PROD SCHEMA로 사원에 대한 관리 업무가 넘어갔습니다.

만약 기존 시스템에서 개발 시 Synonym으로 DB 정보가 제공되지 않고 테이블에 대한 SCHEMA가 제공되었다면, 쿼리를 HR.EMP로 데이터를 조회 했을겁니다.

PROD SCHEMA로 테이블 정보가 넘어가면 기존에 개발되었던 프로그램에서 HR.EMP 로 구성했던 쿼리를 PROD.EMP로 모두 변경해야하는 번거로움이 생깁니다.

만약 EMP라는 Synonym을 제공했다면, 번거로운 프로그램 변경없이 Synonym에 대한 정보만 수정하면 해결 가능합니다.

운영되고 있는 시스템이라면 더더욱 Synonym의 활용이 유용할 것 같습니다

Synonym 생성

GRANT CREATE PUBLIC SYNONYM TO HR;

SYNONYM 생성에 대한 권한이 없다면 DBA 권한의 계정에서 권한을 부여해줍니다.

CREATE 생략가능[PUBLIC] SYNONYM Synonym명 FOR 객체;

CREATE PUBLIC SYNONYM EMP FOR EMPLOYEES;

PUBLIC 옵션을 사용해주면 Synonym을 생성한 계정 뿐만 아니라 다른 계정에서도 사용할 수 있습니다.

CREATE PUBLIC SYNONYM DEPT FOR DEPARTMENTS@USA;

DB LINK를 사용해준다면 위와 같이 생성해주면 됩니다.

SELECT \* FROM EMP;

===========================================

**SYSDATE**

.

[사용법]

SELECT SYSDATE FROM DUAL;

Dual 이라는 테이블은 오라클이 설치될때 만들어지는 테이블이며, 산술,날짜처리,계산 등을 하는 기본 임시 테이블입니다.

간단한 날짜 산술도 가능합니다.

SELECT SYSDATE-1 FROM DUAL; (1일전의 날짜/시간을 반환)

SELECT SYSDATE+1 FROM DUAL; (1일후의 날짜/시간을 반환)

결과 포멧을 유저가 원하는 대로 바꿀수도 있습니다.

SELECT TO\_CHAR(SYSDATE, 'HH24:MI:SS') FROM DUAL; (시간만 반환 (24시간포멧), ex> 16:53:43);

SELECT TO\_CHAR(SYSDATE, 'HH:MI:SS') FROM DUAL; (시간만 반환, ex> 04:53:43);

SELECT TO\_CHAR(SYSDATE, 'YYYYMMDD') FROM DUAL; (날짜만 반환, ex> 20140416);

TO\_CHAR 함수는 임의의 데이터를 문자형 데이터타입으로 변경하는 함수입니다.

반환된 날짜 포멧을 유저가 원하는 스타일로 바꿔서 가져올 수 있습니다.

그럼 SYSDATE 함수가 쓰이는 예제를 하나 해 보겠습니다.

TABLE\_SHOP 테이블에서 유통기한(EXP\_DATE char(8))이 오늘부터 3일 이내인 제품의

이름(PRODUCT\_NAME)을 SELECT해 보겠습니다.

SELECT PRODUCT\_NAME ,

EXP\_DATE

FROM TABLE\_SHOP

WHERE EXP\_DATE < (SELECT TO\_CHAR(SYSDATE, 'YYYYMMDD')+3 FROM DUAL)

===========================================

java.sql.SQLException: 부적합한 열 인덱스

이유

들어갈 값이 부적절해서 발생하는 에러

insert into에서 오류가 발생한 문제이다.

===========================================

**seed PDB**

seed PDB를 이용한 Pluggable 데이테베이스 생성

가상화된 데이터베이스 -> 데이터베이스에 데이터베이스가 있는 것 -> 이러한 것을 Pluggable Database라고 한다

CDB -> container DB

Create

Pluggable Database hrpdb

Admin user dba1

identified by password -> PDB(Pluggable DB) 생성 쿼리 -> 가상 데이터베이스를 만들어 달라

버전과 PDB확인

> 접속을 splplus sys as sysdba // sys계정으로 로그

> select name from v$pdbs;

NAME

PDB$SEED

XEPDB1

ORA-01045 : user lacks create session privilege; logon denied

>> 사용자 이름 생성하고 세션 권한을 부여하지 않아서 발생하는 오류

>> 해결방법

> grant create session to 사용자이름;

권한이 부여되었습니다

https://jack-of-all-trades.tistory.com/286

>> 멀티테넌트 개념

- 원격접속하기

SQL> EXEC DBMS\_XDB.SETLISTENERLOCALACCESS(FALSE);

sys -> sysdba

PL/SQL 처리가 정상적으로 완료되었습니다.

===============================================

**Value Type**

CHAR(size[byte|CHAR]) VARCHAR2(size[BYTE | CHAR])

size -> 1byte

CHAR vs. VARCHAR 차이점

만약 id를 저장하려고 하는데 char와 varchar를 각 50으로 지정했다고 가정하자

사용자가 id를 10글자만 입력했다 그러면 char는 10이 저장되고 40이라는 공간이 남는다

반면 varchar는 50의 공간을 10의 크기로 줄여 저장하기 떄문에 가변적인 문자열이 들어오는 데이터의 경우 varchar를 사용하는 것이 좋다

-> varchar는 구분자로 구분하기 때문에 검색을 하기에 번거롭다. 서로 사용목적에 따른 장단점이 있다

NCHAR(size)NVARCHAR(size)

size -> 2 or 3byte

The number of bytes can be up to two time size for AL16UTF16

encoding and three times size for UTF8 encoding

-> NCHAR는 다른 언어를 넣을 때 사용한다

Date '2013-03-02'

Timestamp'2013-09-03 10.35.10.00000'

create table mem

(

id VARCHAR2(50),

pwd VARCHAR2(50),

name VARCHAR2(50),

gender NCHAR(2), -- char(2 char)로 하면 사용가능 char가 문자형태로 입력되길 바란다

-- but char(2 char)를 사용하는 것은 바람직하지 않다 캐릭터로 2개의 문자를 입력받겠다

-- NCHAR(2)로 사용하는 것이 바람직하다 2글자를 사용하겠다는 의미

age NUMBER,

birthday CHAR(10),

phone char(13),

pegdate date

);

select lengthb('한글') from dual;

select lengthb('ab') from dual;

-> 문자열 길이 확인하기

LONG

-> variable-length character strings containing up to 2 gigabytes-1

CLOB

-> 대용량 텍스트 데이터 타입(최대 4기가) -> 많이 사용함

NCLOB

-> 대용량 텍스트 유니코드 데이터 타입(최대 4기가)

NUMBER(정수와 실수를 동시에 처리함)

NUMBER(4) -> 최대 4자로 이루어진 숫자

NUMBER(6,2) -> 소수점 2자리를 포함하는 최대 6자리의 숫자

NUMBER(6,-2) -> 소수점 -2자리에서 반올림하는 최대6자리의 숫자

NUMBER-> NUMBER(38,\*) 맥시멈 38자리

NUMBER(\*,5)-> NUMBER(38,5)

TIMESTAMP -> NLS\_TIMESTAMP\_FORMAT 파라미터에 명시된 값을 따른다

(TIMESTAMP -> 그 시간을 스탬프로 찍어버리는 느낌)

select \* from NLS\_DATABASE\_PARAMETERS;-> 이걸 기준으로

===============================================

alter table mem (modify) id NVARCHAR2(50);

-> 테이블 변경하기

alter table mem (drop column) age;

-> 컬럼 삭제하기

alter table mem (add) email VARCHAR(200);

-> 컬럼 추가하기

--> 수정을 하는 경우는 데이터베이스를 구축하는 초기에 많이 사용하고 후기로 갈수록 많이 사용하지는 않는다

insert into mem values('id','pwd','name','gender',age,'email');

-> 모든 필드 값을 입력하기

insert into mem(id, pwd) values('newlec','111');

-> 원하는 필드만, 원하는 순서대로 입력하기 -> 그럼 빈 공간은 ?? null값이 들어가죠

insert into mem(id, pwd) values('jeons','111');

insert into mem(id, pwd) values('geni', '123');

select id, name, pwd from mem;

select id as 아이디, name, pwd from mem;

-> 별칭사용하기 id as 아이디 로 사용하면 컬럼명을 변경할 수 있다(as는 생략이 가능하다)

-> 빈 공백을 넣고 싶으면 select id "user id", name, pwd from mem;

-> 그럼 " "안에 들어간 문자는 대문자면 대문자로 소문자면 소문자로 출력이 된다

update mem set pwd='111';

-> update하겠다 setting을 pwd를 111로 모두 다

-> 그럼 모든 비번이 111로 변경되기 때문에 매우매우 위험핟

그래서

update mem set pwd='231' where id ='jeons';

-> where로 조건을 주어서 변경해줘야 한다

--> 오라클은 같은 계정이어도 사용자의 접속 정보를 따로 관리하기 때문에 세션별로 관리한다

--> 그래서 같은 계정으로 하나 더 열어도 결과가 다르다

만약 두개 이상의 컬럼을 업데이트하고 싶다면??

update mem set pwd='123', name='손고쿠' where id='jeons';

--> 콤마로 구분자를 해주고 다음 컬럼의 내용을 입력해준다

삭제를 하고 싶다면

delete mem where id='test';

===============================================

**트랜잭션** ?

- 업무 실행단위, 논리 명령 단위

업무적인 단위 : 계좌이체 -->

물리적인 명령어 단위 : LOCK(2. 그 동안 다른 세션이 건드리지 못하도록 락을 건다) update(1. 현재 세션을 위한 임시저장소에서 테스트), update UNLOCK

(만약 a계좌에서 b계좌로 계좌이체를 했는데 a계좌에서는 돈이 빠져나갔지만, b계좌에는 돈이 안들어왔다..

이런 .. 이런 경우가 발생할 수 있으니까 실패할 경우를 생각하는 것 / 두 계좌가 같이 움직여야 한다)

--> 하나의 단위가 실행되는 동안 다른 세션이 접근하면 데이터의 손실과 데이터의 유실이 발생할 수 있기 때문에

락을 걸어서 하나의 단위가 실행되는 동안에는 다른 세션이 접근하지 못하도록 한다

===============================================

업무적인 단위 : 이벤트 게시글 등록 --> 물리적인 명령어 단위 : insert, update

(글을 작성하면 포인트를 주는 이벤트를 할 때)

테이블 스페이스를 만들 때 임시용과 영구용을 만드는데 임시용에 현제 세션을 임시로 저장하고 있는 것이다

commit; -> 커밋 완료(commit하면 모든 세션에서 볼 수 있게된다), 커밋 버튼을 누르든 commit명령어를 입력을 하든

-> 커밋을 하지 않으면 그전 내용은 임시저장소에만 저장되기 때문에 영구용에 저장되지 않는다

(같은 계정으로 동일한 것을 열어도 커밋을 하지 않으면 락이 걸려서 업데이트가 되지 않는다)

COMMIT 명령어와 ROLLBACK 명령어의 장점

- 데이터 무결성이 보장된다.

- 영구적으로 변경하기 전에 데이터의 변경사항을 확인할 수 있다.

- 논리적으로 연관된 작업을 그룹화할 수 있다.

===============================================

**Rollback**

DB 데이터를 관리 중, 실수로 INSERT, UPDATE, DELETE 후, COMMIT을 했는데, 다시 ROLLBACK을 원한다면,

SELECT \* FROM 테이블명

AS OF TIMESTAMP(SYSTIMESTAMP - INTERVAL '원하는시간' MINUTE);

이렇게하면, 원하는 시간 이전의 데이터가 조회가능합니다.

1). 일단 기존의 데이터는 새로 백업을 합니다.

(백업 생략 가능)

2). 변경 데이터는 DELETE를 해줍니다.

(DELETE FROM 테이블명)

3). 기존 데이터로 INSERT를 해줍니다.

INSERT INTO 테이블명 SELECT \* FROM 테이블명 AS OF TIMESTAMP(SYSTIMESTAMP - INTERVAL '원하는시간' MINUTE);

※ 위의 방법은, 테이블 안의 모든 데이터를 삭제 후, 새로 입력하는 방법입니다

===============================================

**DB 산자d**

select pwd+1 as pwd from mem;

-> pwd에 +1을 하고 컬럼명은 pwd로 출력해줘

select 1+'3' num from dual;

-> num 행의 값을 1+'3'=4가 나옴

select 1+'3' num from dual; -- +는 숫자만 연산한다

> 결과 4

select '3' || 10 num from dual; -- 문자열 더하기는 || 으로 처리한다

> 결과 310

select 1+'a' from dual; -- 숫자로 바꿀수 없다는 오류가 뜬다

select name ||CHR(40)||id||CHR(41) name from mem; -- 특수문자를 문자열에 추가할 때는 chr(아스키코드)를 넣어준다

select name ||'('||id||')' name from mem; -- 특수문자도 문자이기 때문에 작은 따옴표로 구분해준다

연산자

= (!=, ^=, <> --> 같지 않다), <, >, >= , <=, is null, is not null

select \* from mem where name='손고쿠'; -- 이름이 손고쿠인 사람(만) 조회하기

-- 비번이 100이 넘는 사람(만) 조회하기

컬럼을 선택하고 레코드를 선택할 수 있다!!

select \* from mem where pwd > 100;

-- phone을 입력하지 않은 사람 조회하기 (phone이 null값인 사람만 조회하기)

select \* from mem where gender is null;

관계연산자

not, and, or, between, in

-- age가 0,1,2 인 멤버를 조회하기

select \* from mem where age between 0 and 2 order by age asc;

select \* from mem where 0<= age and age<=2; -- and로 범위를 나누어서 사용가능

select \* from mem where age in (0,1,2); -- in을 사용하면 연속되지 않은 값들을 뽑아낼 수 있다!!

select \* from mem where age not in (0,2,4); -- not in을 사용하면 특정 값을 제외하고 뽑아낼 수 있다!!

패턴 비교 연산자

LIKE, %, \_

회원 중 '박'씨 성을 조회하기

select \* from where name LIKE'박%'; --> '박'뒤에는 아무거나 와도 상관없다

-- '손'씨 성을 조회하기

select \* from mem where name like '손%'; (=을 사용하면 안된다)

-- '길'씨이고 이름이 외자인 회원 조회하기

select \* from mem where name like '길\_'; --> 언더바(\_)는 자리수를 말한다

-- '손'씨 성을 제외한 회원 조회하기

select \* from mem where name not like '손%';

-- 회원 이름중에 '호'가 들어간 회우너 조회하기

select \* from mem where name like '%호%';

===============================================

**정규식**

정규식을 이용한 패턴 연산

regexlib.com

번호 있는지 확인하기 정규식으로

\d => 0-9를 표현

{} => 반복을 표현

^ => 정규식 시작

$ => 정규식 종료

01[016-9] -\d\d\d\d-\d\d\d\d

01[016-9] -\d{3,4} -\d{4}

^01[016-9] -\d{3,4} -\d{4}$

REGEXP\_LIKE --> where regexp\_like(first\_name, '^set(v|ph)en$');

제목에 전화번호가 포함된 게시글 조회하기

select \* from mem where regexp\_like(title, '^09[01679]-\d{3,4}-\d{4}$);

--> 여기서 주의할 점

^ $를 사용하면 ^부터 시작해서 $까지 내용이 일치하는지 확인하는 것이라서 중간에 있는것이라면 ^ $을 빼주고 사용하면 된다

select \* from mem where regexp\_like(title, '01[01679]-\d{3,4}-\d{4}$);

===============================================

// rownum을 기준으로 자른것이다

select \* from mem where ROWNUM BETWEEN 1 and 4;

// rownum이 8까지 없으면 출력이 안된다

select \* from mem where rownum between 4 and 8;

where rownum >= 1

where rownum > 5 // 이러면 조건에 안맞아서 나오지 않는다

이러한 문제를 해결하는 쿼리문

select \* from -> from절이 먼저 실행된 결과를 가지고 select문을 실행하는 것이다

그럼 괄호안은 더 이상 mem이 아니라 새로운 결과식이 된다

-> 별칭을 쓰지 않고 따로 rownum을 주게 되면 괄호안의 rownum과 where뒤에 오는 rownum은 서로 다른 것이 되기 때문에 결과값이 안나온다

( select rownum num, mem.\* from mem) -> mem의 한정된 컬럼의 모든 것 : mem.\*

where num between 1 and 5;

중복제거

distinct : 한 칼럼에만 사용할 수 있다( 다른 컬럼과 동시에 사용할 수 없다)

select DISTINCT pwd from mem;

===============================================

**sql**

sql -> 데이터베이스관리시스템에게 질의하는 언어 (DDL, DML, DCL)

DBMS -> 데이터베이스를 관리해주는 시스템이다

DBMS가 왜 필요하냐 ?? -> DataBase : 데이터를 모아서 사용하자 / 중복된 것을 솎아내고 데이터를 단일하게 관리하게 하는 것 (중복을 제거해서 결함을 없앤다)

여기서 sql로 db의 중복된 내용을 단일하게 하는데 중요한 것은 DML이다

DML에는 insert, select, update, delete(Create Retrieve Update Delete)가 있다

select에는 컬럼을 선택하거나 컬럼의 연산 또는 as를 통해서 별칭을 줄 수 있었다 -> 필터링 (정규식, LIKE ,distinct)

함수

문자열함수, 숫자함수, 날짜함수, 변환함수, null관련 함수, 집계 함수

문자열 추출함수 substr(문자열,시작위치, 길이)

select substr('hello',1,3) from dual;

select substr('helloworld',1,4) as string from dual; -- 1번부터 4번까지

select substr('hello',3) as string from dual; -- 3번째부터 끝까지 substring이랑 유사함

select SUBSTRB('ABCDEFG',3) from dual; -- 바이트 단위로 자른다

문자열 추출함수 사용 : 모든 학생의 이름과 출생 월만을 조회하세요

// birth에서 3번째부터 2글자만 가져온다

select name, substr(birth,3,2) from members;

회원 중에서 전화번호가 011로 시작하는 회원의 모든 정보를 출력하세요

select \* from member where substr(phone,1,3) = '011';

select \* from member where phone LIKE '011%'; -> 데이터가 많으면 like가 낫다

회원 중에서 생년월이 7,8,9월인 회원의 모든 정보를 출력하세요

select \* from mem where substr(pwd,1,1) between 1 and 2;

// 정수인 경우 between을 사용

select \* from mem where substr(birth,6,2) in ('07','08','09');

// 문자인 경우 in을 사용

전화번호를 등록하지 않은 회원 중에서 생년 월이 7,8,9월인 회원의 모든 정보를 출력

// and 연산자로 묶어서 처리함

select \* from mem where phone is null and substr(birth,6,2) in ('07','08','09');

문자열 덧셈 함수

select concat('홍','길동') from dual;

select '3' || 10 num from dual; -- 문자열 더하기는 || 으로 처리한다

> 결과 310

문자열 트림 함수

select ltrim (' -h- ') from dual; --> 왼쪽 공백

select rtrim (' -h- ') from dual; --> 오른쪽 공백

select trim (' -h- ') from dual; --> 양쪽 다

upper and lower

select lower('ERWEREW') from dual;

select upper('dfadadf') from dual;

회원 아이디가 newlec인 회원을 조회

select \* from member where upper(id) = 'newlec';

문자열 대치 함수

// we를 you로 변환

select replace('where we are','we','you') from dual;

select translate('where we are','we','you') from dual;

> yhoro yo aro

// 글자 순서를 매치해서 변경한다

회원의 이름과 주소를 조회(주소는 빈칸 없이 출력)

select name, replace(address, ' ', '') from member;

===============================================

24강 들을 차례

문자열 패딩 함수

select rpad('hello',10,0) from dual; -- 문자열이 길이 5인데 5이상 빈칸은 0으로 채워라 문자열 오른쪽에

select lpad('hello',10,0) from dual; -- 문자열의 길이가 5인데 5이상 빈칸은 0으로 채워라 왼쪽부터

회원의 이름을 조회하세요(단, 이름의 길이가 3자가 안되는 경우는 이름 오른쪽을 밑줄로 채우기)

select rpad(name,6,'0')from mem; -> 왜 6이냐? 한글은 1글자당 2바이트라서

첫글자를 대문자로 바꾸는 함수

select initcap('the most important thing is..') from dual;

-- 단어마다 첫글자가 대문자로 바뀌어진다

문자열 검색 함수 instr(문자열, 검색문자열, 위치, 찾을 수)

select instr('ALL we need to is just to..' 'to) from dual;

select instr('All we need to is just to ..','to',1,2) from dual;

-- 문자열의 첫번째부터 2번째 to가 오는 위치를 알려줘

회원의 전화번호에서 두번째 대시(-)문자가 존재하는 위치를 알려줘

select instr(phone, '-',1,2)from mem;

회원의 전화번호에서 첫번째 대시문자와 두번쨰 대시 문자간의 간격은?

010-1234-1234

select instr(phone,'-',4)from mem;

select instr(phone,'-',1,2) - instr(phone,'-',1,1) -1 from mem;

회원의 전화번호에서 첫번쨰와 두번째 사이의 국번을 출력

010-1234-1234

select substr(phone,5, instr(phone,'-',1,2) - instr(phone,'-',1,1)-1) from mem;

-- 5부터 시작해서 4글자 가지고 오는 것

-- instr로 빼서 가지고 온다

문자열 길이 얻는 함수

select length('where we are')from dual;

모든 회원의 핸드폰 번호와 번호의 문자열 길이 조회

select phone,length(phone) from mem;

만약 회원의 전화번호 컬럼에 포함된 문자 '-'를 없앤 전화번호의 길이를 출력

select length(replace(phone,'-',''))from mem;

select ascll('a') from dual;

select chr(65) from dual; -- 아스키코드로 사용가능

절대값을 구하는 함수 abs(n)

select abs(35), abs(-35) from dual;

음수/양수를 알려주는 함수 sign(n)

select sign(35),sign(-90),sign(0) from dual;

-> 1 -1 0 이라는 결과가 나온다

숫자의 반올림 값을 알려주는 함수 Round(n,i)

select round(34.1234), round(31,678) from dual;

-> 34 32

select round(34.1234123,2), round(31.671318,1) from dual;

-> 34.12 31.7

-> 소수점 i번째 자리까지 반올림 하고 오른쪽 숫자를 확인하고 반올림한다

숫자의 나머지 값을 반환하는 함수 mod(n1,n2)

select trunc(16/5) 몫, mod(17,5) 나머지 from dual;

-> 몫 3 나머지 2

숫자의 제곱을 구하는 함수와 제곱근을 구하는 함수 power(n1,n2) / sqrt(n)

select power(5,3), sqrt(16) from dual;

-> 125 4

현재 시간을 얻는 함수

select sysdate, current\_date(세션에 설정된 시간을 가져온다 - current, 접속자의 시간), SYSTIMESTAMP(디테일한 시간), current\_timestamp from dual;

세션 시간과 포맷 변경

alter session set time\_zone='-1:0' -> 나라마다 시차가 다르기 때문에 사용함 -> 그럼 current 시간이 변경된다(session을 변경한것이기 때문이다)

alter session set nls\_date\_format = 'YYYY-MM-DD HH24:MI:SS';

-> 20201-09-25 16:36:23

날짜 추출함수 extract(year,month,day,hour,minute,second..)

select exctract (year from sysdate) from dual;

select exctract (month from sysdate) from dual;

select exctract (day from sysdate) from dual;

select exctract (hour from sysdate) from dual;

select exctract (minute from sysdate) from dual;

select exctract (second from sysdate) from dual;

회원 중 가입 월이 1,2,9,12월인 회원만 조회하기

select \* from mem where EXTRACT(month from pegdate) in (1,2,9,12);

날짜를 누적하는 함수 add\_month(날짜, 정수)

select add\_months(sysdate,-1) from dual;

// 등록한지 6개월이 안된 회원 조회하기

// 데이터의 형태가 date로 되어있어야지 조회가능하다

select \* from mem where add\_months(sysdate,-6) < pegdate;

날짜의 차이를 알아내는 함수 months\_between(날짜,날짜)

select MONTHS\_BETWEEN(SYSDATE,TO\_DATE('2021-01-22'))from dual;

// 등록한지 6개월이 안된 회원조회를 이렇게 할 수 있다

select \* from mem where months\_between(sysdate,pegdate) < 6

다음 요일을 알려주는 함수 next\_day(현재날짜, 다음요일)

select NEXT\_DAY(sysdate, '월요일')from dual;

-> 오늘이 토요일이면 다음 월요일은 언제인가를 확인하는 함수

월의 마지막 일자를 알려주는 함수 last\_day(날짜)

select LAST\_DAY(SYSDATE)from dual;

// 다음달 마지막 일자 확인하기

select last\_day(add\_months(sysdate,1)) from dual;

지정된 범위에서 날짜를 반올림하는 round(날짜,포멧) 날짜를 자르는 trunc(날짜,포멧)

select round(sysdate,'cc'), TRUNC(SYSDATE,'cc')from dual; -- cc 백년단위로 반올림

select round(sysdate,'year'), TRUNC(SYSDATE,'year')from dual; -- year 년단위

select round(sysdate,'q'), TRUNC(SYSDATE,'q')from dual; -- q 분기단위

select round(sysdate,'month'), TRUNC(SYSDATE,'month')from dual; -- month 단위

select round(sysdate,'w'), TRUNC(SYSDATE,'w')from dual; -- w 주단위

select round(sysdate,'day'), TRUNC(SYSDATE,'day')from dual; -- 일단위

select round(sysdate,'d'), TRUNC(SYSDATE,'d')from dual; --일단위

select round(sysdate,'hh'), TRUNC(SYSDATE,'hh')from dual; -- 시간단위

select round(sysdate,'mi'), TRUNC(SYSDATE,'mi')from dual; -- 분단위

// 현재 날짜와 to\_date로 문자열을 날짜로 변환해서 반올림하고 날짜 자르기

select sysdate, round(to\_date('2051-09-11'),'cc'), trunc(to\_date('2051-09-11'), 'cc') from dual;

===============================================

**형변환, 포맷**

숫자 - to\_char() -> 문자열 - to\_date() -> 날짜

<- to\_number() - <- to\_char() -

-- number형식을 문자열(varchar2)로 변환 to\_char(number)

select to\_char(123456789,'$99,999,999,999.99') from dual; -- $99,999,999,999.99가 숫자 길이보다 길어야한다

-- 포맷 문자 설명

-- 9 자리수 표현

-- 0 빈자리를 채우는 문자

-- $ 앞에 $표시

-- , 천 단위 구분자 표시

-- . 소수점 표시

select to\_char(1234) || 'geee' from dual;

select to\_char(232324, '00999999')||'afdf' from dual;

select trim(to\_char(1234567,'9,999,999,999')) ||'원' from dual;

date 형식을 문자열(varchar2)로 변환 to\_char(datetime)

select to\_char(sysdate,'yyyy-mm-dd hh24:mi:ss')from dual;

포맷문자설명

yyyy/rrrr/yy/year년도표시 : 4자리/y2k/2자리/영문

mm/mon//month월 표시 : 2자리/영문3자리/영문전체

dd/day/ddth일 표시 : 2자리/영문/2자리st

am/pm오전/오후

hh/hh24시간표시 : 12시간/24시간

mi분표시 : 0-59분

ss초표시 : 0-59초

문자열을 날짜 형식으로 변환하는 함수 to\_date(문자열, 날짜포멧)

select to\_date('1994-10-11','yyyy-mm-dd')from dual;

문자열을 숫자형식으로 변환하는 함수 to\_timestamp(문자열)

select to\_timestamp('1992-12-12 12:12:12', 'yyyy-mm-dd hh24:mi:ss') from dual;

문자열을 숫자형식으로 변환하는 함수 to\_number(문자열)

select to\_number('1993') from dual;

반환 값이 null일 경우에 대체 값을 제공하는 nvl(null, 대체 값)함수

select nvl(age,0) from mem;

nvl에서 조건을 하나 더 확장한 nvl2(입력값, notnull 대체값, null대체값) 함수

// not null이면 100/age를 넣고 null이면 0을 넣고

select nvl2(age, trunc(age/10)\*10, 0) from mem;

두 값이 같은 경우 null 그렇지 않은 경우 첫번째 값 반환 nullif(값1, 값2) gkatn

select nullif(age,19) from mem;

조건에 따른 값 선택하기 decode(기준값, 비교값, 출력값, 비교값, 출력값)

// gender가 남성이면 1을 출력하고 아니면 2를 출력해라

select decode(gender,'남성',1,2) from mem;

select decode(substr(phone(1,3), '011','skt','016','ktf','etc') || ' , ' || phone from mem;

===============================================

**select 문의 구절**

select, from, where, group by, having, order by 순서 암기할 것

정렬 순서

asc / desc

-> order by 뒤에 컬럼이름을 작성하고 asc나 desc를 안적으면 기본적으로 asc가 작동된다

asc : 오름차순 / desc : 내림차순 (외울 때 desc에서 d가 다운이니까 내림으로 외운다)

이름을 기준으로 역순으로 정렬해서 조회하시오

select 열이름 from 테이블 order by 열이름 desc;

select \* from mem order by name desc;

회원 중에서 '박'씨 성을 가진 회원을 조회하시오.(단, 나이를 오름차순으로 정렬)

select \* from mem where name like '박%' order by age asc;

2차 정렬하기

// 순서안에서 또 다른 순서를 만들기

// age로 정렬하고 name으로 정렬하기(age가 동일하면 name으로 정렬이 된다)

select \* from mem order by age desc, name asc;

===============================================

**집계 함수**

sum, min, max, count, avg

// mem 테이블에서 name이 몇개인가

select count(name) from mem; -> null이 들어가 있는지 없는지 파악해야한다. 즉 null값이 아닌 컬럼의 count를 해야한다

select avg(age) from mem;

select sum(score) from mem; 이런식으로 가능

select id, count(name) count from mem GROUP BY id;

-- name을 카운트 할건데 id당 name의 개수를 카운트하고 싶다

-- 집계되었을 때 한 단위로 표현된다

-- 그룹명과 카운트 되는 대상은 포함해도 되지만 나머지 요소가 들어오면 오류가 발생한다

select id, count(name) count from mem GROUP BY id order by count desc;

-- 집계되었을 때 한 단위로 표현된다

-- 그룹명과 카운트 되는 대상은 포함해도 되지만 나머지 요소가 들어오면 오류가 발생한다

-- 제일 먼저 실행되는 것은 from절이다

-- from -> connect by -> where -> group by -> having -> select -> order by 순서대로 작동된다

-- 그래서 select에서 별칭을 지정하면 having절에서 사용할 수 없다

-- 왜냐하면 구동되는 순서가 having이 먼저이기 때문에 having에선 select에서 정한 별칭이 없기 때문이다

회원별 게시글 수를 조회하시오

select id, count(title) count from member group by id having count(title) <= 2 order by count desc;

select, from, group by, having, order by 순서대로 작성해야 하지만

구동은 이런 순서로 된다 from, group by, having, order by, select

그래서 집계함수는 group by에서만 사용 가능하다

그래서 집계함수를 사용할 수 있는 것은 group by 이후에만 사용가능(구동되는 순서에서) 그래서 order by, having, select에서 사용가능

rownum은 where에서만 사용가능

===============================================

32강 들을 차례

select ROWNUM,id,birthday,age from mem order by age;

-- 왜 섞이는가>> rownum은 where절에서 생성되기 때문에 order by를 하게되면 섞이게 된다

-- where 여기서 만든 번호가

-- group by

-- order by 여기서 섞이게 된것이다

-- 정렬하고 rownum을 붙여야 하는 경우가 있다

-- 이때 row\_number()함수를 사용한다

select ROW\_NUMBER() OVER (ORDER BY age) as ro, id, birthday, age from mem;

-- 정렬된 상태에서 일련번호를 붙이고 싶을 때 row\_number() over(order by [정렬할 컬럼])을 사용한다

-- 정렬된 상태에서 랭크를 매기고 싶을때는 row\_number()대신 rank()를 사용한다

-- row\_number는 일련번호이고 rank()는 등수이다 그래서 값이 같으면 공동 등수가 발생한다

select rank() OVER (ORDER BY age) as ro, id, birthday, age from mem;

-- 만약 공동 등수가 발생하면 4 4 이렇게 나오면 원래는 다음등수가 6이다

-- 근데 다음 등수를 5로 하고 싶다면 dense\_rank()를 사용한다

select dense\_rank() OVER (ORDER BY age) as ro, id, birthday, age from mem;

-- id별 작성한 내용의 등수를 확인하고 싶다면 partition by를 사용한다

-- id를 그룹별로 묶어서 내용의 등수를 확인하는 쿼리문

-- 여기서 partition by를 한 id가 오름차순으로 정렬된 것을 확인할 수 있다

-- 여기서 age를 desc하면 age에 대한 desc가 일어나지만 id는 여전히 오름차순으로 정렬이다

select ROW\_NUMBER() OVER (PARTITION BY id ORDER BY age) as ro, id, birthday, age from mem;

구절의 순서를 바꿔야 하는 경우

select \* from member where rownum between 1 and 10;

-> 최신 등록순으로 정렬한 결과에서 상위 10명을 원하는 경우라면???

select \* from member order by pegdate desc where rownum between 1 and 5;

-> 순서 떄문에 오류가 발생한다

그럼 해결 방법은

select \* from (select \* from member order by pegdate desc) where rownum between 1 and 5;

-> from절에 정렬된 결과를 넣어서 다시 rownum으로 자른다

-> from절에는 반드시 테이블이 와야하는 것이 아니라 격자형 데이터가 와도 된다

-> 괄호가 서브쿼리를 나타낸다

select \*

from (select \* from mem order by age desc)

where rownum between 1 and 5;

-> 이런식으로 사용가능하다

평균 나이 이상인 회원 목록을 조회하시오

select \*

from mem

where age >=

(select avg(age) from mem);

-> select avg(age) from mem으로 평균나이를 구하고 where절로 평균나이보다 큰 회원을 조회한다

===============================================

**DBMS의 개요**

- 데이터의 무결성(데이터가 수정되거나 바뀌지 아니한다)

- 데이터의 중복을 없애준다(각각의 베이스별로 사용되던 데이터를 한곳에 모아서 관리)

- 실시간으로 데이터를 업데이트하면서 싱크발생하는 것을 없앨 수 있다

- 데이터마다 공통된 부분을 데이터베이스에 저장하기 때문에 다시 원래의 데이터로 돌아갈 때는 공통부분에 공통되지 않는 부분을 합쳐줘야 한다

- 참조하고 있는 데이터를 원래의 데이터로 합치는 작업을 조인이라고 한다

===============================================

**조인의 종류**

- inner join, outer join, self join, cross join(Cartesian Product)

join의 예

(1 : N)을 의미한다, 까마귀발

-> 부모 자식 테이블은 상대적인 것이다

-> 자식 테이블의 title에서 댓글을 다는 기능을 구현하면

-> 자식 테이블이 부모테이블이 되는 것이다

-> writer\_id가 id를 외래키로 참조하고 있다

( 부모테이블 )( 자식 테이블)

id name \_\_\_\_/\_idtitlewriter\_id

newlec뉴렉 \1oracle 개요newlec

2SQL구문newlec

3DML사용하기newlec

위의 테이블을 합치면

id name idtitlewriter\_id

newlec뉴렉 1oracle 개요newlec

newlec뉴렉2SQL구문newlec

newlec뉴렉3DML사용하기newlec

-> id멤버가 글을 등록한 관계가 있는 것이다

이런 형식의 테이블이 된다 (이게 원본이었는데 공통 부분인 id,name을 분리하고 id, title, writer\_id를 자식 테이블로 생성한 것이다)

-> 왜 이렇게 할까??

-> 확실히 밑에 테이블보다 위에 테이블이 간결하다

-> 부모테이블의 중복된 내용를 없애버렸기 때문이다

-> 예시에서는 중복된 데이터가 2개이지만, 데이터의 양이 커지면 많은 양의 중복데이터를 없앨 수 있기 때문에 성능이 좋아진다

-> 즉, 디스크에서 데이터를 가지고 올 때 io작업이 줄어들기 때문에 성능이 좋아지는 것이다

select \*

from member inner join notice -- member테이블이 회원의 정보가 있는 테이블이고 notice테이블이 게시글을 저장하는 테이블이다

on member.id = notice.writer\_id; -- 여기서 on 명령어로 명시해줘야 할 것이 있다. member.id와 notice.writer\_id가 동일하다는 것을

-- 즉, writer\_id가 외래키로 member의 id를 참조한다는 것이다

-- member의 id를 기준으로 보았을 때 notice에서 member의 id로 게시글을 작성한 것이 있는 id하고만 join하겠다

-- 합칠 때 관계가 없는 id는 빠지게 된다

-- 자식이 참조한 수만큼의 부모를 복제하게 되고 그 다음 합치게 되는 것이다

-- 관계가 있는 레코드들만 합치는 조인을 inner join이라고 한다

-- 서로 관계가 있는 레코드들은 inner, 관계가 없는 레코드들은 outer

참조키를 기준으로 일치하는 행만 조인

member inner join notice on member.id = notice.writer\_id

참조키를 기준으로 일치하지 않는 행도 포함시키는 조인

member left/right/full outer join notice

on member.id = notice.writer

-> 왼쪽의 outer를 포함하겠다면 left

-> 오른쪽의 outer를 포함하겠다면 right

-> 양쪽 다 포함하겠다면 full

idname \_\_\_\_\_/\_seqtitlewriter

newlec뉴렉 \1oracle개요newlec

yjs유재석2sql구문newlec

jung정형돈3dml사용하기newlec

kang강호동4dml사용하기dragon(결함/ 부모가 없는데 게시글이 있기 때문에)

5dml사용하기dragon

left를 했을 경우

idname seqtitlewriter

newlec뉴렉 1oracle개요newlec

newlec뉴렉2sql구문newlec

newlec뉴렉3dml사용하기newlec

yjs유재석nullnullnull

jung정형돈nullnullnull

kang강호동nullnullnull

-> 이런식으로 자식 테이블의 게시글 수 만큼 부모 테이블읠 복사하고 부모 테이블에서 게시글 데이터가 없는 부분은 null로 채운다

-> 실무에서는 inner보다 outer를 더 많이 사용한다

===============================================

**조인된 테이블 쿼리하기**

id nameseqtitlewriter\_id

newlec뉴렉1oracle개요newlec

yjs유재석2sql구문newlec

jung정형돈3dml사용하기newlec

kang강호동4dml사용하기dragon

5dml사용하기dragon

게시글의 번호, 제목, 작성자 이름을 조회하시오

select seq, title, name

from member inner join notice on member.id = notice.writer\_id

필드 이름의 충동문제

-> 게시글의 컬럼명을 식별하기 위해 테이블명을 사용

-> 컬럼명이 같은 테이블을 조인할 때

select id, name, name from member inner join notice on member.id = notice.writer\_id

-- name이 중복되기 때문에 어느 테이블의 name인지 식별하지 못한다

해결방법

select member.id, member.name, notice.name from member inner join notice on member.id = notice.writer\_id

컬럼을 지정하는 식별자를 줄이기 위해서 테이블의 별칠을 사용

select n.id n.writer\_id, m.name

from member m inner join notice n

on m.id = n.writer\_id;

-> from이 먼저 실행되니까 from에서 별칭을 지정해주면 select에서 사용 가능하다

id, name 그리고 회원별 작성한 게시글 수를 조회하시오

select m.id, m.name count(n.id) count

from member m inner join notice n

on member.id = notice.writer\_id group by m.id, m.name;

--> 그럼 작성한 게시글이 0인 회원이 하나도 없다? 이건 좀 이상하다

--> 게시글은 없지만 작성자를 나오게 해야 한다 그럼 left outer join 사용

select id, name, count(n.id)

from member m left outer join notice n

on member.id = notice,writer\_id group by m.id, m.name;

-> 주인공쪽으로 방향을 잡아서 outer조인해준다

self join (자기 조인)

-> 데이터가 서로 포함 관계를 가지는 경우 : 담당, 구성, 연락

-> 자기가 자기랑 합쳐지는 경우 (하나의 테이블이 2개인것처럼해서 합쳐지는 것)

-> 자기를 관리하는 사람이 있는 경우(부서장)

idnameboss\_idboss\_name

newlec뉴렉nullnull

yjs유재석newlec뉴렉

jung정형돈yjs유재석

kang강호동newlec뉴렉

-> 확장하고자 하는 데이터를 자기자신에게 컬럼을 조인하는 것이다

select m.\*, b.name boss\_name

from mem m left outer join mem b on b.id = m.boss\_id;

-- 댓글이 댓글을 참조하는 경우

-- 카테고리가 카테고리를 참조하는 경우

-- ansi inner join

select n.id, n.title, m.name

from member m inner join notice n on m.id = n.writer\_id

where m.id = 'newlec';

--Oracle inner join

select n.id, n.title, m.name

from member m, notice n

where m.id = n.writer\_id and m.id = 'newlec';

--> 같은 결과가 나온다

--ansi outer join

select n.\*, m.name writer\_name

from notice n left outer join member m

on m.id = n.writer\_id;

--Oracle outer join

select n.\*, m.name writer\_name

from notice n, member m

where m.id(+) = n.writer\_id; -> 여기서 (+)가 left랑 동일한 의미

--> Oracle에서는 full outer join을 지원하지 않는다 즉, 양쪽에 (+)부호를 붙일 수 없다는 말이다

Cross join(관계 상관없이 두개의 테이블을 곱하여 결과를 나타내는 것)

-- ansi cross join

select n.\*, m.name writer\_name

from member m cross join notice n

--> 업무에서 잘 사용하지 않는다

유니온과 조인

union으로 합치는 방법(관련이 없어도 합칠 수 있다, 주의할 점은 양쪽 테이블의 컬럼의 개수만 맞춰주면 된다)

-> 3개의 게시판을 하나의 게시판으로 합치고 싶을 때 사용된다 -> 통합검색

레코드를 합치는 방법

1. join으로 합치는 방법

select n.id, n.title, m.name

from member m inner join notice n

on m.id = n.writer\_id

2. union으로 합치는 방법

select id, name from member

union

select writer\_id, title from notice;

union 연산자

notice

idtitlewriter\_id

1oraclenewlec

2sqlnewlec

3dml2newlec

Qna

idtitlewriter\_id

1oraclenewlec

2sqlnewlec

3dmlnewlec

->> union(중복을 제거한다) ,

minus (qna에서 notice와 중복되는 내용을 제거하고 notice를 출력 -> 3 dml2 newlec ),

intersect (qna와 비교해서 공통분모를 살리는것 notice에서 1과 2가 출력),

union all(중복허용하고 전부 합치는 것)

===============================================

**view (뷰) 만들기**

-> 테이블을 넓게 보거나 좁게 보아야할 경우가 있다

물리적인 데이터구조(table)와 개념적인 데이터 구조(view)의 차이

-> table은 무결성으로 fk로 서로 연결되어 있는데 이것을 view를 통해 원하는 데이터를 뽑아서 볼 수 있게 하는 것이다

idname \_\_\_\_/\_idtitlewriter\_id \_\_\_/\_ idcommentnotice\_id

newlec뉴렉 \1oraclenewlec\ 1 많은 도움1

yjs유재석2sqlnewlec2정리굿1

jung정형돈3dmlnewlec3감사합니다3

kang강호동4dmldragon

5dmldragon

select n.seq, n.title, n.writer, m.name, count(c.seq) cnt from

member m

inner join notices n on m.id = n.writer

inner join comments c on n.seq = c.noticeseq

group by n.seq, n.title, n.writer, m.name;

--> 이건 올바른 방법이 아니다

가운데 테이블을 중심으로 간다

select n.id, n.title, n.writer\_id, m.name, count(c.id) cnt from

member m

right outer join notice n on m.id = n.writer\_id-- 오른쪽하고 outer조인하고

left outer join comment c on n.id = c.notice\_id -- 왼쪽하고 outer조인하기

group by n.id, n.title, n.writer\_id, m.name;

이러한 쿼리문을 view로 정의해서 사용하기

-> 마치 테이블로 정의되어 있는 것처럼 사용하기

create view noticeview

as

select n.id, n.title, n.writer\_id, m.name, count(c.id) cnt from

member m

right outer join notice n on m.id = n.writer\_id-- 오른쪽하고 outer조인하고

left outer join comment c on n.id = c.notice\_id -- 왼쪽하고 outer조인하기

group by n.id, n.title, n.writer\_id, m.name;

-> 이렇게 view를 만들 수 있다

-> 즉, view는 말 그대로 창을 보는 것이다

-> 보고자하는 데이터를 미리 만들어두고 view로 정의해서 보는 것이다

view를 이용해 쿼리하기 (마치 테이블처럼 사용가능)

-> 목록에 있는 내용을 가져올 때 뷰로 만들어서 사용가능

-> select \* from noticeview

-> select \* from noticeview where titiel like '%오%'

order by seq desc;

===============================================

**데이터 딕셔너리**

-> oracle DBMS에 사용자의 정보, 권한, 테이블, 뷰, 제약조건, 함수/프로시저등을 담고 있는 딕셔너리를 말한다

-> 데이터가 테이블형태로 제공하고 있는데 그걸 직접사용할 수 없게 해놓았다

-> 단일 테이블에서 사용자에 보여주고 싶지 않은 컬럼을 제외하고 사용자에게 보여줄 수 있다

-> 제외하고자 하는 컬럼에 제한을 걸어서 볼 수 없게 해버릴 수 있다

-> 단일 테이블에서 view를 사용하면 너무 낭비이기 때문이다

-> select \* from dict;

-> 딕셔너리 뷰를 확인할 수 있다

DBA\_ , ALL\_, USER\_ -> (tables, tab\_columns, all\_tables, tab\_comments, col\_comments, external\_locations, tab\_histograms, tab\_statics)

select \*from user\_tables;

select \* from user\_tab\_columns where table\_name ='member';

-> 이런식으로 쿼림문으로 확인할 수 있다

42강 들을 차례

데이터가 제약조건없이 들어가게되면 유효하지 않은 데이터가 들어갈 수 있다. 그럼 데이터의 무결성에 위배된다. 제약조건을 걸어서 데이터의 결함이 없도록 무결성을 유지하도록 한다.

도메인 제약조건

- 도메인 : 컬럼의 유효한 값의 범위

- 속성에 도메인이 아닌 값이 올 수 없도록 하는 제약조건

- not null, default, check

- ora-01400 : null을 안에 삽입할 수 없습니다 -> not null 조건을 걸어줬을 때 null값을 넣으면 발생하는 오류

테이블 생성 한 후에 not null 적용 방법

-> alter table test modify email varchar2(200) not null;

굳이 직접 입력할 필요가 없는 컬럼들 : 등록일(default sysdate), 조회수, 관리자가 부여하는 비번

-> 테이블 생성 한 후에 default 적용방법 : alter table test modify regist\_date date default sysdate;

-> timestamp 면 systimestap를 사용

테이블 생성 한 후에 check 제약 조건 적용 방법

-> alter table test add constranint ck\_test\_phone check(phone like '010-%-\_\_\_\_'); -> 이런식으로 like로 제약조건을 걸수있다

- ora-02290 : 체크 제약조건이 위배되었습니다 라고 경고문이 뜬다

정규식을 이요한 체크 제약조건

-> 휴대폰 번호 입력 받을 경우 010-\_\_\_\_-\_\_\_\_

regexp\_like(phone, '010-\d{4}-\d{4}') -> 이렇게 만들어 줄 수 있다

-> ^ : 시작 , $ : 종료 가 포함되어야지 확인할 때 좋다

-> why ? ^ $을 넣지 않으면 010-\d{4}-\d{4}의 내용이 어디든 들어가 있으면 만족하기 때문에

-> 010으로 시작하고 \d{4}로 끝나는 문장안에서 판별할 수 있도록 해야한다

엔티디 제약조건

primary key 와 unique : 중복된 레코드가 없도록 제한한다. 식별 컬럼에 중복이 발생하지 않도록 관리하는 것

primary key : unique + not null

unique : 고유키, 널은 허용하나 중복은 안됨

primary key

-> sequence

create sequence student\_seq

start with 1

maxvalue 99999999999

increment by 1

nocycle -> max찍고 다시 1로 돌아올 것이냐 안할것이냐

nocache; -> 숫자를 미리 20개를 뽑아놔라

student\_seq.nextval : 다음 값

student\_seq.currval : 현재 값

릴레이션 제약조건

===============================================

**Programmers SQL 문제**

1. ANIMAL\_INS 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다.

2. ANIMAL\_INS 테이블 구조는 다음과 같으며, ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, INTAKE\_CONDITION, NAME, SEX\_UPON\_INTAKE는

각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

3. ANIMAL\_OUTS 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다. ANIMAL\_OUTS 테이블 구조는 다음과 같으며,

ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, NAME, SEX\_UPON\_OUTCOME는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다.

4. ANIMAL\_OUTS 테이블의 ANIMAL\_ID는 ANIMAL\_INS의 ANIMAL\_ID의 외래 키입니다.

-> animal\_outs.animal\_id = animal\_ins.animal\_id

5. 입양을 간 동물 중, 보호 기간이 가장 길었던 동물 두 마리의 아이디와 이름을 조회하는 SQL문을 작성해주세요.

-> animal\_outs.date - animal\_ins.date 해서 큰 2개의 데이터를 가지고 오면 된다

즉 두개의 테이블에서 date를 뺀 값을 desc한다

--> 별칭은 from에서 지정해 준다

--> from, group by, having, order by, select 순서대로 읽기 때문에 from에 지정하면 뒤에서 사용 가능

--> from -> connect by -> where -> group by -> having -> select -> order by 순서대로 작동된다

--> 그래서 select에서 별칭을 지정하면 having절에서 사용할 수 없다

--> 왜냐하면 구동되는 순서가 having이 먼저이기 때문에 having에선 select에서 정한 별칭이 없기 때문이다

1단계 시간을 뺀 값이 큰 순서대로 정렬하기 (외래키 사용)

select animal\_outs.animal\_id, animal\_outs.animal\_name, animal\_outs.date - animal\_ins.date as time

from animal\_outs , animal\_ins

where animal\_out.animal\_id = animal\_ins.animal\_id

order by time desc;

2단계 테이블에 별칭을 주어 간결하게 만들기

select a.animal\_id, a.animal\_name, a.date - b.date as time

from animal\_outs a , animal\_ins b

where a.animal\_id = b.animal\_id

order by time desc;

3단계 rownum으로 3이하인 row를 출력한다(서브 쿼리를 사용해서 정렬한 내용에서 가져오고 싶은 부분을 자른다)

select a.animal\_id, a.name

from (select a.animal\_id, a.name, a.datetime - b.datetime as time

from animal\_outs a , animal\_ins b

where a.animal\_id = b.animal\_id

order by time desc)a

where rownum < 3;

===============================================

1. ANIMAL\_INS 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다.

2. ANIMAL\_INS 테이블 구조는 다음과 같으며, ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, INTAKE\_CONDITION, NAME, SEX\_UPON\_INTAKE는

각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

3. ANIMAL\_OUTS 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다.

4. ANIMAL\_OUTS 테이블 구조는 다음과 같으며, ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, NAME, SEX\_UPON\_OUTCOME는 각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다.

5. ANIMAL\_OUTS 테이블의 ANIMAL\_ID는 ANIMAL\_INS의 ANIMAL\_ID의 외래 키입니다.

--> on animal\_outs.animal\_id = animal\_ins.animal\_id

6. ANIMAL\_OUTS 테이블에서

- Allie의 ID는 ANIMAL\_INS에 없으므로, Allie의 데이터는 유실되었습니다.

- Gia의 ID는 ANIMAL\_INS에 있으므로, Gia의 데이터는 유실되지 않았습니다.

- Spice의 ID는 ANIMAL\_INS에 없으므로, Spice의 데이터는 유실되었습니다.

--> animal\_outs에서 allie와 spice의 데이터를 뽑아내면 되는 거다

--> 주인공쪽으로 방향을 잡아서 outer조인해준다

--> 여기서 주인공은 animal\_outs이다

--> 그럼 animal\_outs left outer join animal\_ins 라고 지정해 주면 된다

--> animal\_outs.animal\_id 와 animal\_ins.animal\_id 가 외래키이니까 동일하게 가지고 있을건데

--> animal\_ins쪽에서 빈 값이고 animal\_outs쪽에는 있는 값을 가지고 오면 된다

주인공은 animal\_outs

select animal\_outs.animal\_id, animal\_outs.name

from animal\_outs left outer join animal\_ins

on animal\_outs.animal\_id = animal\_ins.animal\_id

where animal\_ins.animal\_id is null

order by animal\_outs.animal\_id;

--> 결론은 animal\_outs에서 값을 가져와야 하기 떄문에 animal\_outs left join animal\_ins 를 사용해서 포커스를 outs에 맞춤

--> 그리고 animal\_outs.animal\_id와 animal\_ins.animal\_id가 외래키인 것을 이용해서

--> animal\_ins.animal\_id가 null값은 것을 animal\_outs.animal\_id에서 찾아서 가지고 오는 것이다

===============================================

1. ANIMAL\_INS 테이블은 동물 보호소에 들어온 동물의 정보를 담은 테이블입니다.

2. ANIMAL\_INS 테이블 구조는 다음과 같으며, ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, INTAKE\_CONDITION, NAME, SEX\_UPON\_INTAKE는

각각 동물의 아이디, 생물 종, 보호 시작일, 보호 시작 시 상태, 이름, 성별 및 중성화 여부를 나타냅니다.

3. ANIMAL\_OUTS 테이블은 동물 보호소에서 입양 보낸 동물의 정보를 담은 테이블입니다.

4. ANIMAL\_OUTS 테이블 구조는 다음과 같으며, ANIMAL\_ID, ANIMAL\_TYPE, DATETIME, NAME, SEX\_UPON\_OUTCOME는

각각 동물의 아이디, 생물 종, 입양일, 이름, 성별 및 중성화 여부를 나타냅니다.

5. ANIMAL\_OUTS 테이블의 ANIMAL\_ID는 ANIMAL\_INS의 ANIMAL\_ID의 외래 키입니다.

--> on animal\_outs.animal\_id = animal\_ins.animal\_id

6. 관리자의 실수로 일부 동물의 입양일이 잘못 입력되었습니다.

보호 시작일보다 입양일이 더 빠른 동물의 아이디와 이름을 조회하는 SQL문을 작성해주세요.

이때 결과는 보호 시작일이 빠른 순으로 조회해야합니다.

--> animal\_outs.datetime < animal\_ins.datetime

--> ins가 더 큰 값을 찾아내는 것

--> from animal\_outs left join animal\_ins -- 조인을 하는데 외래키 조건이 필요하다

on animal\_outs.animal\_id = animal\_ins.animal\_id -- on 조건으로 외래키를 지정해준다

select a.animal\_id, a.name

from animal\_outs a left join animal\_ins b

on a.animal\_id = b.animal\_id

where a.datetime < b.datetime

order by b.datetime;

**브릿지로 공용 DB만들기**

1. 리눅스를 클릭하고 오른쪽 마우스로 setting에 들어간다

2. Network Adapter 를 bridge로 설정

3. 그리고 service oracle-xe-18c restart로 restart

4. vi /etc/sysconfig/network-scripts/ifcfg-ens33

에 들어가서 bootproto에서 static을 dhcp로 변경해서 주소를 받아온다

5. ifconfig로 현재 주소를 받아온다

6. 그리고 받아온 주소를 vi /etc/sysconfig/network-scripts/ifcfg-ens33에

들어가서 IPADDR에 받아온 주소를 넣어준다

7. 그리고 service network restart 해준다

8. ifconfig로 들어가서 확인해보면 변경된 주소를 확인할 수 있다

===============================================

**sql 새 계정 만들기**

1. su - oracle

2. . oraenv

3. sqlplus / as sysdba로 들어간다

4. sql> create user c##mango identified by it; 로 새 계정 생성

5. sql> grant connect, resource, dba to c##mango; 로 권한을 준다

-> 이렇게 하면 새로운 계정이 생성됨

===============================================

**sql developer에서 접속하는 방법**

1. developer에 접속해서 새로운 접속을 연다

2. 그리고 첨부한 사진처럼 입력을 한다

3. 아이디는 c##mango 비번은 it

4. ip는 221.164.9.200 이고 포트번호는 50000으로 해서 테스트 해보고 접속이 되는지 확인한다

5. 테스트가 성공한 것을 확인하면 접속을 하면 공용DB로 접속이 가능하다

===============================================

**​oracle db io 오류**

오라클 데이터베이스 에러 (상태: 실패 -테스트 실패: IO 오류: The Network Adapter could not establish the connection)

디바이스의 이름을 변경해서 발생한 오류로 OracleXETNListener가 실행이 정지되었을 가능성이 높다 이것을 시작으로 변경해 준다.

이 때는 net Configuration Assistant 에 들어가서 새로운 리스너를 만들어서 호스트 네임을 변경해주는 것이 좋다.

listener refused the connection with the following error ora-12505

12505 오류는 DB의 SID 이름이 잘못되어서 그런 것이다.

oracle lsnrctl status -> cmd 창에 lsnrctl status 를 입력하면 현재 상태를 확인할 수 있다.

===============================================

**oracle의 리스너는 ???**

Oracle : 리스너(Listener)

오라클 리스너는 네트워크를 이용하여 클라이언트에서 오라클 서버로 연결하기 위한 오라클 네트워크 관리자 입니다.

네트워크를 통한 연결은 모두 리스너가 담당하며 리스너와 연결되기

위해서는 클라이언트에 오라클 NET8(오라클 클라이언트)이 설치되어 있고 이를 통해

오라클 서비스명이라는 것을 만들어 접속을 해야 합니다.

오라클 서버에서 리스너를 시작시켜줘야 클라이언트들이 접속을 할 수 있습니다.

lsnrctl 명령어로 리스너를 관리 할 수 있습니다

**<windows cmd콘솔에서 명령>**

c:\>lsnrctl help -> 도움말

c:\>lsnrctl status -> 현재 상태

c:\>lsnrctl start-> 시작

c:\>lsnrctl stop -> 죽이기

c:\>lsnrctl reload -> 재시작

또는

c:\>lsnrctl ->관리모드 실행

**<관련파일>**

c:\>oracle\ora92\network\admin

listener.ora / tnsnames.ora 파일이 있다. 만약 내가 서버라면 두개의 파일이 모두 있고, 클라이언트라면 tnsnames.ora 파일만 있는 것을 확인 할 수 있다.

oracle - client 간의 네트워크 설정을 하기 위한 파일로써.

서버 - listener.ora // 클라이언트 - tnsnames.ora 을 설정.

listener.ora -> 오라클 서버에서 클라이언트의 요청을 듣고, 클라이언트와의 통신 환경을 설정하는 파일. 오라클 클라이언트에서 서버로 접속할 때 플요한 프로토콜(주로TCP/IP) 및 포트 등을 설정

tnsnames.ora -> 클라이언트에서 서버로 접속할 때 필요한 프로토콜, 포트, 서버주소, 인스턴트등을 설정함. 클라이언트에 위치