

Oracle



1. Oracle 설치

1. <http://www.oracle.com> 에 접속하여 Oracle 다운로드(11g)

2. <http://sqlgate.com/> 에 접속하여 Oracle 개발 Tool 다운로드

3. <http://www.oracle.com/t> Oracle Sql Developer Tool 다운로드

주의

- 컴퓨터 계정명 확인하기(한글안됨)
- 설치경로 한글경로 안됨

3. 알아두기

Oracle XE 버전에는 SCOTT 계정이 미리 준비되어 있지 않고 또 EMP 테이블도 생성되어 있지 않은 상태로 배포되고 있기 때문에 학습용으로 사용하기에 좋은 환경을 구성하기 위해서는 설치 후에 SCOTT 계정과 EMP 테이블을 생성해 주어야 한다.

EMP 테이블은 생성된 상태가 아니지만 설치 폴더 안에는 scott.sql 파일이 포함되어 있으며 관리자 계정에서 scott.sql 파일을 실행하면 scott 계정이 생성되고 SCOTT 계정에 속하는 모든 디폴트 테이블 등이 생성된다



3. 알아두기

1. SCOTT.SQL 파일 실행

```
SQL> conn system/[passwd];
```

```
SQL> @C:\oracle\app\oracle\product\11.2.0\server\rdbms\admin\scott.sql
```

2. SCOTT 계정으로 접속

```
SQL> conn scott/tiger;
```

3. SCOTT 계정이 잠겨있을 경우

```
SQL> ALTER USER scott ACCOUNT UNLOCK;
```

4. SCOTT 계정의 암호모를때 관리자 계정으로 접속하여 SCOTT의 암호를 변경

```
SQL> ALTER USER scott IDENTIFIED BY tiger;
```



3. 알아두기

5. SCOTT 계정을 새로 생성해야 한다면 관리자 계정으로 접속하여 SCOTT 계정을 생성

```
SQL> CREATE USER scott IDENTIFIED BY tiger;
```

6. SCOTT 계정으로 접속이 거부된다면, 관리자 계정으로 접속하여 권한을 부여

```
SQL> GRANT RESOURCE TO scott;
```



3. 알아보기

① system 비밀번호 모를 경우 접속방법
=> cmd창을 열어서 sqlplus 엔터.
sys as sysdba;
=> 암호 입력은 Enter

② 비밀번호 변경
alter user 계정명 identified by 비밀번호;

③ 접근 가능한 table 목록 => select * from tab;

④ 특정테이블에 컬럼에 대한 정보 => desc table이름;

⑤ 접속된 상태에서 다른 계정의 접속 변경할 때
conn 계정명/비밀번호;

⑥ 현재 접속된 user 알아보기
show user;



4. 사용자 계정 만들기

- 최고 관리자 인 system 계정으로 접속하여 만듦

① 계정 만들기

create user 계정명 identified by 비밀번호;

② 만든 계정의 lock 풀기

alter user 계정명 account unlock;

ALTER USER 계정명 IDENTIFIED BY 비밀번호 ACCOUNT UNLOCK;

③ 기본 권한설정(create table / insert/ update/ delete /select)

grant connect, resource to 계정명;

④ 권한 주기(특정 권한 부여)

grant create session, create table, create view,
create sequence, create procedure
to 계정명;

5. SQL의 종류

DQL	Data Query Language - 데이터검색(select문장)
DML	Data Manipulation Language - 데이터 변경/ 추가(Insert / Update / Delete)
DDL	Data Definition Language - DataBase구조정의(Create / Alter / Drop / Rename)
DCL	Data Control Language - 권한지정, 제거(Grant / Revoke)
TCL	Transaction Control Language) 논리적인 작업의 단위를 묶어서 DML에 의해 조작된 결과를 작업단위(transaction)별로 제어하는 명령어 (commit, rollback)

6. DataType(자료형)

문자형	<p>char(byte수) => 2000byte 최대지정 -고정길이</p> <p>varchar2(byte수) => 4000byte 최대지정 -가변길이</p> <ul style="list-style-type: none">* char 타입에 byte수를 지정하지 않으면 기본 1byte* 문자는 반드시 ' ' 묶는다. <p>Nchar / NVarchar2 => 유니코드 문자형 데이터타입</p> <p>Long - 가변길이 문자형 (최대 크기는 2GB)</p>
숫자형	<p>number(전체자리수, 소수점자리수)</p> <p>ex) number(2) - 정수 2자리 (-99 ~ 99)</p> <p>number(5, 2) - 전체 5자리 중에 2자리수가 소수점자리</p> <p>int</p>
날짜형	<p>date</p> <ul style="list-style-type: none">* '년-월-일' or '년/월/일'* 현재날짜와 시간을 가져오는 함수 - sysdate

7. 테이블 생성

생성방법

```
create table 테이블이름(  
    컬럼명 datatype [ null | not null ] [ constraint 별칭 제약조건 ] ,  
    컬럼명 datatype [ null | not null ] [ constraint 별칭 제약조건 ] ,  
    .....  
)
```

주의

- * [] 기호는 생략가능의 의미
- * | 기호는 or 의미

8. 제약조건 종류

① Primary Key

```
CREATE TABLE userlist(  
  id VARCHAR2(10) CONSTRAINT id_pk PRIMARY key,  
  name varchar2(10) not null  
)
```

특징

- 기본키, 중복안되고 not null 임.
- 하나의 테이블에 기본키는 반드시 한 개 만 존재.

8. 제약조건 종류

② Foreign Key

```
CREATE TABLE fk_member(  
    code number(2) NOT NULL ,  
    id VARCHAR2(20) NOT NULL  
        CONSTRAINT id_fk REFERENCES MEMBER(id) ,  
    etc VARCHAR2(10)  
)
```

특징

- 다른 테이블의 기본키를 참조하는 키
- 중복가능/ null허용
- 참조되고 있는 테이블의 데이터 값 이외의 값은 삽입할 수 없음.
(insert 할 때 잘못된 데이터 삽입이 안되도록 하는 것.)

8. 제약조건 종류

② Foreign Key

- 테이블 생성시 fk를 지정할 때 cascade 옵션을 설정하여
- 부모레코드가 삭제될 때 자식 레코드가 함께 삭제되게 할 수 있다.

```
create table userlist(  
    id varchar2(20) primary key,  
    age number(2)  
);
```

```
create table book(  
    no number(2) primary key,  
    id varchar2(20)  
    constraint book_id_fk references userlist(id) on delete cascade  
)
```

8. 제약조건 종류

③ Unique

```
CREATE TABLE userlist(  
  id VARCHAR2(10) CONSTRAINT id_pk PRIMARY key,  
  jumin char(13) not null CONSTRAINT jumin_un unique  
)
```

특징

- 중복 안됨. (유일한 값)
- null을 허용함(만약, not null 지정하면 null허용 안됨)
- 테이블을 만들 때 pk와 같은 효과를 주기 위한 제약조건.
- 한 테이블에 여러 컬럼에 사용가능 함.

8. 제약조건 종류

④ check

```
CREATE TABLE ck_Test(  
  NAME VARCHAR2(10) NOT NULL,  
  age NUMBER(2) NOT NULL  
    CHECK (age BETWEEN 20 AND 30 )  
    --age컬럼의 값은 20~30사이만 허용  
)
```

특징

- 조건을 주어 조건에 만족하는 값 만 삽입 가능하도록하는 것.

8. Default (기본값 지정하기)

default

```
CREATE TABLE de_Test(  
  NAME VARCHAR2(10) NOT NULL,  
  addr VARCHAR2(10) DEFAULT '서울'  
)
```

특징

- 기본값을 지정하여 값을 넣지 않아도 기본값이 들어가도록 함.
- null을 허용했을때 default컬럼의 값을 생략하면 무조건 default가 들어감.
- null을 입력하고 싶을때는 직접 null을 넣어야 null이 들어감.

9. 테이블 수정 / 삭제

① 컬럼추가

```
alter table 테이블이름 add  
    (컬럼명 자료형 [제약조건] , 컬럼명 자료형 [제약조건] , ....)
```

② 컬럼삭제

```
alter table 테이블이름 drop column 컬럼이름
```

③ datatype변경

```
alter table 테이블이름 modify 컬럼이름 변경자료형
```

④ 컬럼이름 변경

```
alter table 테이블이름 rename column 기존컬럼명 to 변경컬럼명
```

⑤ 테이블 삭제

```
drop table 테이블이름
```

9. 테이블 이름 수정

①테이블 이름 변경

rename 변경전테이블이름 to 변경후테이블이름;

10. 레코드 삽입 / 수정 / 삭제

레코드 삽입

insert into 테이블이름(컬럼명, 컬럼명,...) values(값, 값, ...)

insert into 테이블이름 values(값, 값, ...)

=> 모든 컬럼에 모두 값을 넣을 때 사용

레코드 수정

update 테이블이름

set 컬럼명=변경값 , 컬럼명=변경값 , 컬럼명=변경값 ,

[where 조건식]

레코드 삭제

Delete table이름 [where 조건식]

truncate table 테이블이름 ; => 모든 레코드 삭제

11. 레코드 검색

레코드 검색

```
select distinct | * | 컬럼명 as 별칭, 컬럼명 별칭,....  
from 테이블이름  
[where 조건식 ]  
[order by 컬럼명 desc | asc , .. ]
```

예

- 중복행제거(distinct)

```
select addr from member; -- 모든 레코드 검색(중복포함)
```

```
select distinct addr from member; -- 중복된 addr는 한번만 검색
```

- 컬럼명에 별칭주기(as or 공백)

```
select id as 아이디 , name "이름" from member;
```

- 정렬

```
SELECT * FROM EMP ORDER BY job asc , sal desc;
```

12. 연산자

① 산술연산자

+	더하기
-	빼기
*	곱하기
/	나누기
mod	나머지 ex) Mod(값, 나눌 수)

② 관계연산자

>	크다
<	작다
>=	크거나 같다
<=	작거나 같다
!=	같지않다
=	같다

알아
두기

- nvl 함수

nvl(값, 대치값) - 값에 null이 있으면 대치값으로 변경한다.

12. 연산자

③ 조건연산자

And	그리고 (양쪽모두 true일 때 만족)
Or	또는(양쪽 중 한 개라도 true이면 만족)
컬럼명 In (값, 값, ..)	ex) id in (1,3,5) => id가 1 또는 3또는 5
컬럼명 Between 최소 and 최대	ex) age between 10 and 20 => age의 값이 10~20 사이
컬럼명 like '와일드카드문자열'	*와일드카드 _ => 한 글자 검색 % => 0개이상의 글자 검색 ex) name like 'j%' ; name like '____' ; name like 'j_a%' ;

12. 연산자

③ 조건연산자

any	- 검색결과와 하나 이상이 일치하면 참 ex) 컬럼명 < any(100, 200, 300) => 최대값보다 작다 컬럼명 > any(100, 200, 300) => 최소값 보다 크다
all	- 검색결과와 모든 값이 일치하면 참. ex) 컬럼명 < all(100, 200, 300) => 최소값보다 작다 컬럼명 > all(100, 200, 300) => 최대값보다 크다
not	not => 현재상태의 반대값 ex) id not in (1,3,5) age not between 10 and 20 name not like 'j%'

④ null 값 찾기

- is null => null인 값 찾기
- is not null => null 아닌 값 찾기

13. 함수

① 집계함수

- sum(컬럼명) => 합계
- avg(컬럼명) => 평균(null값은 제외하고 나눔)
- max(컬럼명) => 최대값
- min(컬럼명) => 최소값
- count(컬럼명) => 총 레코드수(null값은 제외함)
- count(*) => null을 포함한 총 레코드수
- rank(expr) within group(order by 컬럼명 asc | desc)
=> 전체 값을 대상으로 각 값의 순위를 구함.

ex) --급여가 3000의 등수 구하기

```
SELECT RANK(3000) within GROUP(ORDER BY sal desc)  
FROM EMP;
```

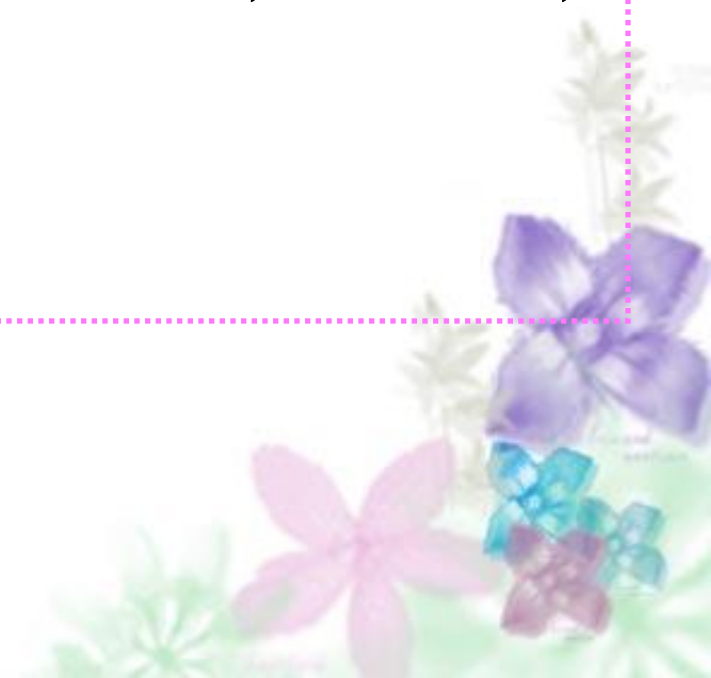

13. 함수

집계함수 사용시 Group by

- 집계함수를 이용할 때 group by에 없는 일반 컬럼명은 함께 검색할 수 없다.
- 특정컬럼 기준으로 group by 를 하여 각 그룹별 집계함수를 사용 수 있다.

ex) -- 각 반 별 kor의 총점, 평균, 최대, 최소

```
SELECT ban , SUM(kor) 합계, AVG(kor), MAX(kor), MIN(kor) ,  
COUNT(*) , COUNT(kor)  
FROM sungjuk  
GROUP BY ban;
```



13. 함수

② 숫자함수

- $\text{round}(\text{숫자}, \text{자리수}) \Rightarrow$ 반올림
- $\text{ceil}(\text{숫자}) \Rightarrow$ 올림 한 후 정수반환
- $\text{floor}(\text{숫자}) \Rightarrow$ 내림 한 후 정수 반환
- $\text{mod}(\text{숫자}, \text{나눌수}) \Rightarrow$ 나머지
- $\text{trunc}(\text{숫자}, \text{자리수}) \Rightarrow$ 버림



13. 함수

③ 문자열 함수

- upper(문자열) => 모두 대문자
- lower(문자열) => 모두 소문자
- initcap(문자열) => 단어의 첫 글자 대문자로 표현
- length(문자열) => 문자열의 길이
- substr(문자열, 시작, 개수) => 문자의 일부분 추출
- instr(문자열, 찾을문자열) => 찾을 문자열의 출현 위치 알려줌
- instr(문자열, 찾을문자열, 시작번지수, 몇번지 위치)
ex) instr(job, 'A', -1) => -1은 뒤에서 부터 검색.
- lpad(문자열, 전체자리수, 특정문자)
=> 오른쪽 정렬 후 왼쪽 빈 공백에 특정문자로 채움
- Rpad(문자열, 전체자리수, 특정문자)
=> 왼쪽 정렬 후 오른쪽 빈 공백에 특정문자로 채움)
- ltrim() => 왼쪽 공백제거
- rtrim() => 오른쪽 공백제거

13. 함수

④ 날짜함수

- sysdate => 현재날짜와 시간
- months_between(날짜, 날짜) => 두 날짜 사이의 개월 수를 구함
- add_months(날짜, 숫자) => 날짜에서 숫자만큼 월을 더함.
- next_day(날짜, 요일) => 날짜에서 가장 가까운 요일의 날짜구함
(1 = 일요일 , 2 = 월요일,.....)
- last_day(날짜) => 날짜 달의 마지막 날짜를 구함.

13. 함수

⑤ 형변환 함수

- to_char(datetime, format형식)=> 날짜를 format형식으로 변환
ex) to_char(sysdate, 'YYYY-MM-DD')
to_char(sysdate, 'YYYY-MM-DD HH:MI:SS')
to_char(sysdate, 'YYYY-MM-DD HH24:MI:SS')
- to_char(number, format형식)=> 숫자를 format형식으로 변환
ex) 3자로 콤마
to_cahr(2000, '999,999') => 2,000
to_cahr(2000, 'L999,999') => \2,000 => 지역에 따른 화폐표시
to_cahr(2000, '\$999,999') => \$2,000
- to_date(문자열) => 문자를 날짜로 변환
- to_number(문자열)=> 문자를 숫자로 변환

13. 함수

- select절에서 조건을 주어 조건에 따라 실행문장을 다르게 할 때 사용하는 함수

```
decode(대상,  
      값, 문장,  
      값, 문장,  
      값, 문장,,,,,)
```

=> 대상에 해당하는 값이 일치하는 경우 사용함.

```
case [대상]  
  when 조건1 then 문장  
  when 조건2 then 문장  
  when 조건3 then 문장  
  ...  
  else 문장  
end
```



14. Join

두 개 이상의 테이블을 하나의 테이블로 만들어 한번의 검색으로 여러 컬럼의 정보를 확인할 때 사용함.

종류

- inner Join
- outer Join
 - left outer Join
 - right outer join
 - full outer join
- self join



14. Join

Inner Join 방법

```
select 컬럼명 , .....  
from 테이블이름 별칭 [Inner] join  테이블이름  별칭  
on  별칭.컬럼명 = 별칭.컬럼명
```

- 여러개의 테이블 조인

```
select 컬럼명 , .....  
from 테이블이름 별칭 [Inner] join  테이블이름  별칭  
on 별칭.컬럼명 = 별칭.컬럼명  [Inner] join  테이블이름  별칭  
on 별칭.컬럼명 = 별칭.컬럼명 ...
```



14. Join

outer left join 방법

```
select 컬럼명, .....  
from 테이블이름 별칭 outer left join 테이블이름 별칭  
on 별칭.컬럼명 = 별칭.컬럼명
```

outer right join 방법

```
select 컬럼명, .....  
from 테이블이름 별칭 outer right join 테이블이름 별칭  
on 별칭.컬럼명 = 별칭.컬럼명
```

outer full join 방법

```
select 컬럼명, .....  
from 테이블이름 별칭 outer full join 테이블이름 별칭  
on 별칭.컬럼명 = 별칭.컬럼명
```



14. Join

Self join – 자기 자신과 조인 하는 것

```
select 컬럼명, .....  
from 테이블이름 별칭 join 테이블이름 별칭  
on 별칭.컬럼명 = 별칭.컬럼명
```

Ex) 각, 사원 별 매니저는 누가?

```
SELECT empno, ename, mgr FROM EMP;
```

```
select a.EMPNO 번호 , a.ename 이름 , b.empno , b.ename  
FROM EMP a join EMP b -- a는 일반사원 정보, b는 매니저 정보  
on a.mgr = b.empno --사원정보의 mgr이 매니저의 empno가 같은 것
```



15. Set 집합

합집합

union - 여러개의 sql문의 결과에 대한 합집합으로 결과에서 모든 중복된 행은 하나의 행으로 만든다.

union all - 여러개의 sql문의 결과에 대한 합집합으로 중복된 행도 그대로 결과로 표시한다.

교집합

intersect - 여러개의 sql문의 결과에 대한 합집합. 중복된 행은 하나의 행으로 만든다. (조인과 동일하다)

차집합

minus - 앞의 sql문의 결과에서 뒤의 sql문의 결과에 대한 차집합이다. 중복된 행은 하나의 행으로 만든다.

15. Set 집합

주의

- select문장의 컬럼의 개수와 데이터타입이 일치해야 한다.
(char(5) 와 varchar(5) 은 다르게 본다)
- order by절은 맨 마지막에 한번 사용가능 함.



15. Set 집합

union


- 두 개의 테이블의 레코드를 합치는 것.(중복 행 제거함)

```
SELECT * FROM emp  
UNION  
SELECT * FROM c_emp;
```

Union all

- 두 개의 테이블의 레코드를 합치는 것.(중복 행 제거 안함.)

```
SELECT * FROM EMP  
UNION all  
SELECT * FROM C_EMP;
```



15. Set 집합

intersect

```
SELECT product_id FROM INVENTORIES  
INTERSECT  --교집합  
SELECT product_id FROM ORDER_ITEMS
```

minus

```
SELECT product_id FROM INVENTORIES  
MINUS  -- 차집합  
SELECT product_id FROM ORDER_ITEMS
```



16. SubQuery

- ① 메인 쿼리 안에서 또 다른 쿼리문이 있는 것.
- ② 반드시 서브쿼리를 괄호로 묶는다.
- ③ 메인쿼리 보다 서브쿼리가 먼저 실행되어 결과를 메인 쿼리의 조건으로 사용
- ④ 서브쿼리의 결과가 한 개 이상 반환될때는 in , any, all 연산자를 사용함.
- ⑤ 서브쿼리의 결과가 한 개일 때는 비교연산자 사용함.

ex) select * from emp where ename = (서브쿼리문장) ;

17. View

- ① 가상테이블
- ② 실제 존재하지 않지만 실제 테이블과 똑같이 사용함
(select , insert , delete, update 가능)
- ③ 복잡한 쿼리문을(조인, 서브쿼리)미리 뷰로 만들어 사용하면
간단하게 검색 가능함.(단,insert , delete, update 안됨)
- ④ 보안을 위해 사용함(관리자 유형에 따라 특정 컬럼 만 선택하여 보여줌)

17. View

View 생성 방법

```
create view 뷰이름  
as  
  뷰의 내용
```

View 수정 방법

```
create or replace view 뷰이름  
as  
  뷰의 내용
```

View 삭제 방법

```
drop view 뷰이름
```

18. 테이블 복사 / 레코드 복사

테이블 복사

=> 이미 생성된 테이블의 구조나 레코드와 열을 선택적으로 복사할 수 있다.

=> 방법

```
create table 테이블이름  
as  
복사할테이블내용
```

레코드 복사

=> insert 할 때 레코드를 다른 테이블에서 가져와서 삽입하기

=> 방법

```
insert into 테이블이름  
select 컬럼명, ... from 테이블이름
```

18. 테이블 복사 / 레코드 복사

테이블 복사 예)

- emp테이블의 모든컬럼과 모든 레코드 복사하기
=> `CREATE TABLE c_emp AS SELECT * FROM EMP;`
- emp테이블의 특정컬럼을 복사하기(컬럼명 이름 변경포함)
`CREATE TABLE c_emp2 AS
SELECT empno 번호, ename 이름, job 직업, sal 급여 FROM EMP;`
- emp테이블의 특정레코드만 복사하기(조건사용)
`CREATE TABLE c_emp3 AS
SELECT * FROM EMP WHERE deptno=20;`
- emp테이블의 테이블의 구조만 복사하기(조건을 항상 불만족표현)
`CREATE TABLE c_emp4 AS SELECT * FROM EMP WHERE 1=0`

19. rownum

- 검색할 때 자동 행 번호 추가

ex)

```
SELECT ROWNUM, ename, sal  
FROM emp;
```

ex)

```
SELECT ROWNUM, ename, sal  
FROM (SELECT * FROM EMP ORDER BY sal)
```



20. Sequence

- 자동증가 컬럼.
- 순차적으로 값을 증가하여 내부적으로 유일한 값을 만듦
- 시퀀스를 생성할 때 시작값 , 최대값 , 증가치를 지정하여 자동증가 하도록 만듦

Sequence 생성방법

```
create sequence 시퀀스이름 -- 1부터 시작  
[start with 시작값 ]  
[increment by 증가치 ]  
[maxvalue 최대값 ]  
[minvalue 최소값 ]  
[cycle | nocycle ]  
[cache | nocache ]
```

20. Sequence

Sequence 사용법

- 시퀀스이름.nextval => 시퀀스값 증가
- 시퀀스이름.currval => 현재 시퀀스값 가져오기

Sequence 수정

alter sequence 시퀀스이름

Sequence 삭제

drop sequence 시퀀스이름

21. transaction

- insert ,update ,delete를 수행한 결과를 저장(commit), 취소(rollback) 할 수 있다.
- 특정 영역을 나누어서 저장, 취소를 선택적으로 할 수 있다
savepoint 이름; =>SQL실행 전에 영역을 분할함.
rollback 이름; => 특정 영역에 해당하는 부분까지 취소됨.
- Commit - 전체저장완료
- rollback - 전체취소

22. 다른계정에 있는 테이블 접근하기

- 현재 접속된 user 알아보기
show user;
- 다른계정에 있는 테이블에 접근하기 위해서는 권한설정 필요.
 1. system계정으로 접속하여 권한부여
grant all on 계정명.테이블이름 to 계정명;
 2. 다른 계정으로 접속
conn 계정명/비번
 3. 다른 계정에 있는 테이블접속하기
SELECT * FROM 계정명.테이블이름;
 4. 권한 취소
revoke all on 계정명.테이블이름 from 계정명;

23. 데이터베이스 백업하기

1. 테이블을 export 하는 방법.

=> data를 옮기고 싶은 테이블을 dmp 파일로 만든다.

방법) cmd 창을 연다(cmd창 위치는 상관 없다.)

```
exp id/pwd tables=(xxx,xxx,xxx ...) file=c:\aaa.dmp
```

```
exp id/pwd file=c:\aaa.dmp -> 모든테이블 백업
```

=> xxx부분은 테이블이름으로 가지고 오고자하는 테이블 이름을콤마로 연결한다.

=> aaa.dmp는 aaa는 원하는 파일명지정한다. c:\폴더에 만들어진다.

2. 만들어진 aaa.dmp 파일을 import 하는 방법.

방법) cmd 창을 연다.(cmd창 위치는 상관 없다.)

```
imp id/pwd ignore=y full=y file=c:\aaa.dmp
```

```
imp id/pwd file=c:\aaa.dmp full=y
```

=> c:\aaa.dmp는 export로 만들 어진 파일을 가지고와
현재 컴퓨터의 c:\폴더에 넣어놓아야한다.

=> 기존에 테이블이 존재 한다면 데이터가 추가 되고
테이블이 존재 하지 않으면 테이블을 자동으로 만들어 추가된다.

감

사

함

니

다

!