Database & SQL

-데이터베이스 : 구조화된 정보(data) == 자료를 그룹화
-데이터베이스의 특징
공통, 통합, 저장,운영 데이터의 집합
1. 공용데이터 -> 공통
2. 통합테이터 -> 통합, 최소한중복, 통제된중복
2. 저장데이터 -> 디스크나 저장매체
4. 운영데이터 -> 조직에 운영에 반드시 필요한 데이터 저장

예) 학사 관리 시스템
교무과: 전반적인 행정
학적과: 입학,휴학
학생과: 학생 정보

DB(DBMS)
1. 실시간 접근성: 실시간 처리 응답
2. 계속적인 변화: 갱신,삽입,삭제(동적)
3. 통시 공용: 사용자가 동시 사옹
4. 내용에 의한 참조: 위치나 주소가아닌 값에 따라참조 데이터페이스 검색

DBMS(Database Management System): -> DB
응용프로그램 연결 DB-> DBMS
DB+DBMS+응용프로그램 -> DBMS

-> RDBMS -> 관계형 데이터베이스 시스템 -. oracle, Mysql ,Access,,, H2(test용)
-> SQL(Structure Query Language) -> 관계형 데이터 베이스 프로그램 CRUD

관계형 데이터 모델 SQL
구조(Stricture) -> 객체타입, 관계 * 릴레이션(테이블)
연산(Operation) -> 관계대수, 데이터 조작
제약조건(Constraint)-> 무결성 제약조건

**ERD -> 데이터 모델링
개체 관계 다이어그램
E(Entity) - R(Relastion) -D(Diagram)

데이터 모델링
개념적모델링
논리적모델링
물리적모델링

** 데이터베이스 생성 -> 테이블 생성 -> 테이블간에 관계설정 -> CRUD
xe create table 테이블명 join설정

1.테이블 생성 member

2.테이블(member) -> column(열) , row(행)

릴레이션 (테이블(2차원), 튜플(행row)의 집합)
테이블의 핵-> 튜플 ***** MemberDto
도메인-> 속성(필드)의 범위 ***** MemberDto

*릴레이션의 특징
1. 튜플의 유일성(동일한 릴레이션을 가질 수 없다)
2. 집합은 중복을 허용하지 않음
*튜플의 무순서성: 튜플은 순서가 없다
3. 속성의 무순서성->
4. 속성의 원자성-> 속성의 값은 더이상 분해 할 수 없다.

**제약 조건

*****
키(Key): 하나의 테이블 내에서 각 튜플의 유일하게 식별 할 수 있는 속성등의 집합

- 후보키(Candidate key) -> 기본키가 될 수 있는 키
  유일성 , 최소성
  학번이름주소 학번
- 슈퍼키(Super key)-> 학번-이름, 학번=이름-주소,결합
  유일성을 반족하는 키, 최소성 만족 하지 않아도 됨
- 기본키(Primary key) -> 유일성, 중복 불가능 -> 주민번호 ,학번,아이디, ******
  ***** 개체 무결성 -> 후보키 중에 주키
- 대체키(Altemative key) -> 후보키 중 기본키를 제외한 나머지 후보키
- 외래키(FK): R1, R2의 연관 관계를 설정 ->
  *** foreign key 참조무결성

1. 개체무결성 -> 기본키는 null을 가질 수 없다. 중복불가능
2. 참조무결성-> 외래키값은 피참조 릴레이션의 기본키이거나 null값이다
3. 도메인무결성-> 속성값은 속성 도메인(범위)에 속한 값들 중 하나 이어랴한다.

***무결성 위반

```
create table member(
        userId varchar2(100) ,
        userPw varchar2(100) not null,
        age number(3) not null,
        primary key(userId)
);
```

SQL -> RDMS
1. SQL의 역사와 특징
◆SQL의 역사● SEQUEL(Structured English Query Language)
▪ 1974년, IBM San Jose Lab(현재 IBM Almaden 연구소)

SQL의 개념
1. SQL의 역사와 특징
◆SQL의 특징
1 SQL은 무엇인가?▪ 종합 데이터베이스 언어 ⇨ 데이터 정의(DDL), 조작(DML), 제어(DCL)
2. SQL 기본 구문
◆DDL : 데이터 정의문 create, drop, alter -> 데이터베이스, 테이블 생성, 수정, 삭제
◆DML : 데이터 조작문 -> 데이터 CRUD -> 프로그래머가 가장 많이 사용
insert, update, delete, select
◆DCL : 데이터 제어문 -> 제어, 권한
**사용자권한: grant, revoke, deny
TCL : transaction -> commit(변동사항을 저장), rollback(변동사항 이전으로 돌리기)
commit(변동사항을 저장)-> 추가(insert),수정(update), 삭제(delete) -> oracle -> commit명령어를 추가;
rollback -> 실행 한단계 뒤로
**** 자동 commit -> create, drop, alter, grant, revoke

** 1. 설계 -> 2. 관계설정
1. DDL(데이터정의: Date Definition Language) create, drop alter,truncate
creat -> 데이터베이스,테이블, 사용자, 뷰, 시컨스,인덱스, 생성
drop -> 데이터베이스,테이블, 사용자, 뷰, 시컨스,인덱스, 삭제
alter -> 테이블의 속성, 제약조건(constraint),인덱스, 이름 수정 -> 구조
열추가: alter table 테이블이름 add 열이름 데이터타입
열데이터타입변경: alter table 테이블이름 modify 열이름 데이터타입
열삭제: atler table 테이블이름 drop 열이름

테이블 이름변경: rename table 이전테이블이름 to 새로운테이블이름/
alter table 이전테이블이름 rename to 새로운테이블명;

alter table 테이블명 rename column 이전컬럼이름 to 새컬럼명; **

truncate -> 테이블의 행 삭제

```
alter table member1 add email varchar2(100);--열 추가
alter table member1 modify email varchar2(200);--열 수정
alter table member1 drop(email) ; --열삭제
alter table member1 DROP COLUMN email; --열삭제

--테이블 생성
create table member1(
        userId varchar2(20) not null ,
        userPw varchar2(20) not null,
        age number(3) not null,
    email varchar2(100) not null,
        primary key(userId)
);
```

```
alter table member1 add email varchar2(100);--열 추가
alter table member1 modify email varchar2(200);--열 수정
alter table member1 drop(email)  ; --열삭제
alter table member1 DROP COLUMN email; --열삭제

desc member1;
--rename table member1 to member11 ; --테이블 이름을 변경
alter table member1 rename to member11; --테이블 이름을 변경
desc member11;
commit;

alter table member11 rename to member1; --테이블 이름을 변경
desc member1;
commit;
이름       널?         유형
------ -------- -------------
USERID  NOT NULL VARCHAR2(20)
USERPW  NOT NULL VARCHAR2(20)
AGE     NOT NULL NUMBER(3)
EMAIL   NOT NULL VARCHAR2(100)

2. DML(Data Manipulation Language, 데이터 조작어)
응용소프트웨어 개발자가 주로 사용

- 데이터: insert, delete, update, select
insert  -> 데이터 추가   C  -> insert into  ~ values;
          insert into 테이블이름(column1, cloumn2, ...) values (value1, value2, ...);
          insert into 테이블이름 values(value1, value2, ...)
select  ->  데이터 조회  R
          select 칼럼명 from 테이블;-- 테이블의  칼럼명 칼럼을 조회
           select username from dab_users;
           select *  from dab_users; -- 테이블에서 모든 칼럼의 레코드 조회
update ->데이터 수정   U  -> 조건에 맞는 컬럼의 레코드를 수정
          update  테이블   set  칼럼1=변경값,칼럼2=변경값,,,  where 컬럼명=조건;
delete  -> 데이터 삭제  D
          delete  from 테이블명 where 컬럼명=조건;  --칼럼명이 조건에 맞는  레코드를 삭제


3. DCL(Data Control Language, 데이터 제어어): (권한)
grant : 권한 생성
revoke: 권한 삭제



*사용자 생성
create user 사용자명 identified by 계정비밀번호;
```

create user users2 identified by 1111;

* show user; --접속 계정

*사용자 권한
//권한부여
grant connect, resource to 사용자명; << 관리자
grant connect, resource to user2;
connect-> 접속권한
resource-> 객체및 조작권한
dba -> 관리자 권한
grant connect, resource ,dba to user2;

commit ; /*변경 사항 저장*/

show user; /*현재 접속 계정*/

select username from dba_users;

select * from all_users; /*user 조회*/

//권한 삭제
revoke connect, resource from 사용자명
//계정 삭제
drop user 계정명 cascade;

exit; 종료

**sqldeveloper ->
한줄 복사 ctr+shift+D
한줄 삭제 alt+shift+D

1. 사용자 생성
create user 계정명(아이디) identified by 비빌번호;
create user user2 identified by 1111;
2. 사용자 권한
//권한부여
grant connect, resource to 사용자명; -- 관리자 sys ,system
grant connect, resource to user2;
connect-> 접속권한
resource-> 객체및 조작권한
dba -> 관리자 권한
commit ; /*변경 사항 저장*/

show user; /*현재 접속 계정*/
select username from dba_users; --관리자 권한 조회
select * from all_users; /*user 조회*/

//권한 삭제
3. revoke connect, resource from 사용자명
//계정 삭제
4. drop user 계정명 cascade;
exit; 종료

4. TCL(Transaction Controller Language): 트랜잭션 제어어
** 데이터의 일관성, 동시발생을 보장
COMMIT
ROLLBACK
SAVEPOINT -> 취소 지점

트랜잭션(Transaction): 작업 하나의 단위
COMMIT-> 작업 확정
ROLLBACK-> 작업 취소

오라클 데이터 타입

문자 데이터 타입
 CHAR(n) 고정길이 문자 / 최대 2000byte / 디폴트 값은 1byte **
 VARCHAR2(n) 가변길이 문자 / 최대 4000BYTE / 디폴트 값은 1byte **
 NCLOB 대용량 텍스트 유니코드 데이터 타입(최대 4Gbyte)

VARCHAR2(10) >>> 문자열 가변형 5 -> 자동으로 VARCHAR2(5)
Char(10) >>> 문자열 고정형 5 -> CHAR(10)

숫자 데이터 타입
 NUMBER(P,S) 가변숫자 / P (1 ~ 38, 디폴트 : 38) / S (-84 ~ 127, 디폴트 값 : 0) / 최대 22byte
**
 NUMBER -> 자동으로 크기 조절 **
날짜 데이터 타입
DATE BC 4712년 1월 1일부터 9999년 12월 31일, 연, 월, 일, 시, 분, 초 까지 입력 가능 **
TIMESTAMP 연도, 월, 일, 시, 분, 초 + 밀리초까지 입력가능

RDBMS종류
Oracle, Mysql,SQLServer, Access, H2

NoSql DB -> SQL를 사용하지 않는다..
몽고DB(MongoDB) -> {key:value,key:value}

---

1. 제약조건 (constraint)
데이터무결성
NOT NULL - NULL 값 허용 불가 (값을 반드시 추가)
UNIQUE - 하나의 테이블 내에서 한번만 나옴 - 주로 대체키 설정 시 사용됨
PRIMARY KEY- 기본키(UNIQUE + NOT NULL)
FOREIGN KEY - 외래키 -> 다른테이블의 키를 참조하는 키
CHECK- 도메인 무결성

--테이블 생성 --
```
create table member(
userId varchar2(100) not null, --제약조건
userPw varchar2(100) not null,
age number(3) not null,
primary key(userId)
);
```
-제약조건의 설정
테이블을 만들 때 속성에 제약조건 지정하기
CREATE TABLE 테이블명(
속성명 속성타입 [[제약조건명] 제약조건],
)
*** 참조 무결성 제약조건- 외래키 값은 다른 테이블의 기본키 값들 중에 하나이어야 함
속성명 [CONSTRAINTS 제약조건명] REFERENCE 참조테이블명(속성명)

2. 제약조건 변경
제약조건의 추가 및 제거
테이블을 생성한 후에 제약조건을 추가하거나 제거할 필요성이 있음 제약조건도 테이블의 구조 정보에 속함으로 ALTER TABLE을 이용함
- 추가 : ADD CONSTRAINT - 제거 : DROP CONSTRAINT
제약조건 이름을 지정해 두어야 추가나 제거가 쉬움 - 추가
- 제거
ALTER TABLE 테이블명ADD [CONSTRAINT 제약조건명] 제약조건 (속성명)
ALTER TABLE 테이블명

1. 간단한 데이터 검색
제약조건의 설정
무조건 검색
*- 모든 속성명을 쓰기 힘들 경우 사용- SELECT절에서 *는 모든 속성이란 의미임
AS 키워드- AS 키워드로 속성명을 부여함
간단한 조건 검색
WHERE 절 비교 연산자
- 같다 : = -같지않다 :!=,<> - 크다 : >- 크거나 같다 : >=- 작다 : <- 작거나 같다 : <=
논리 연산자- WHERE 절에서 여러 개의 조건을 결합할 경우 - X AND Y : X, Y가 참일 때 참을 반환- X OR Y : X나 Y가 참일 때 참을 반환- NOT X : X가 거짓일 때 참을 반환

page18image3832224.png ¬ page18image3830208.png ¬ page18image3750304.png ¬
SEL

DISTINCT
SQL은 Bag을 기반으로 함 - 중복된 것들도 다 제시 됨
중복된 것을 제거하고 한번만 나오게 하려면? - DISTINCT를 사용함

2. 복잡한 데이터 검색
BETWEEN, IN, IS NULL
BETWEEN a AND b- 검색 조건의 상한과 하한을 지정함
IN(a, b, c,...)- 속성값이 a, b, c ,... 중 하나라도 일치하면 참
IS NULL- NULL 값은 어떤 비교를 하든 거짓임
문자열 검색
LIKE 연산자
- 컬럼에 저장된 문자열 중에서 LIKE 연산자에서 지정한 문자 패턴과 부분적으로 일치하면 참이 되는 연산자
ORDER BY 절
- 질의문의 결과는 테이블에 입력된 순서대로 출력
- 데이터의 출력 순서를 특정 속성값을 기준으로 오름차순 또는 내림차순으로 정렬해야 하는 경우가 자주 발생함
ORDER BY {column_name} [ASC|DESC]
ASC : 오름차순으로, 기본값 (생략가능) DESC : 내림차순, 생략불가능

1. INSERT 절
다양한 INSERT 구문
단일행 입력- 한번에 하나의 튜플을 테이블에 입력하는 방법
NULL의 입력- 데이터를 입력하는 시점에서 해당 속성값을 모르거나, 미확정일 때 사용함 - NOT NULL 조건이 지정된 경우 입력이 불가능함
서브 쿼리를 이용한 데이터 삽입- 한번에 여러 튜플을 넣을 수 있음
질의 결과 테이블 만들기- 질의 결과로 만든 테이블은 기존 테이블의 속성명과 타입을 그대로 적용함 - NOT NULL 조건을 그대로 적용함- 다른 제약조건은 적용되지 않음
테이블 구조의 복사- 상황에 따라서 기존 테이블과 동일한 구조를 지니는 테이블을 생성할 필요가 있음
테이블의 구조 검색문 - 오라클
DESCRIBE[DESC] 테이블명
- MS SQL
sp_help 테이블명
VALUES를 이용한 다중행 입력- MS-SQL 2008 부터는 서브 쿼리가 아닌 VALUES를 이용해서도 다중행 삽입이
가능함

2. UPDATE구문
데이터 수정
데이터 수정- UPDATE 문은 테이블에 저장된 데이터를 수정하기 위한 데이터 조작어임
서브 쿼리를 이용한 데이터 수정- UPDATE문의 SET절에서 서브 쿼리를 이용함- 다른 테이블에 저장된 데이터를 검색하여 한번에 여러 속성값을 수정할 수 있음 - SET절의 속성명의 서브 쿼리의 속성명과 달라도 됨
복수 속성값 변경- SET 절에 (속성명 = 값), (속성명= 값), ... 으로 작성

데이터 삭제- DELETE 문의 테이블에 저장된 데이터 삭제를 위한 조작어
서브쿼리를 이용한 데이터 삭제- WHERE 절에서 서브 쿼리를 이용함- 다른 테이블에 저장된 데이터를 검색하여 한번에 여러 행을 삭제함 - WHERE 절의 속성명은 서브 쿼리의 속성명과 달라도 됨

*** 트랜잭션(Transaction)이란? -> spring boot JPA @Transanctional
- 트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 수행 되어야할 일련의 연산들
- 트랜잭션은 작업의 완전성을 보장 , 작업 단위. -> 현금 인출기
- 트랜잭션은 SELECT, UPDATE, INSERT, DELETE와 같은 연산을 수행하여 데이터베이스의 상태를 변화시키는 작업의 단위다.

***트랜잭션의 특징(ACID)
- Atomicity(원자성)
    - 트랜잭션이 데이터베이스에 모두 반영되던지, 아니면 전혀 반영 되지 않아야 한다.
    - 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 한다.
- Consistency(일관성)
    - 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다.
    - 시스템이 가지고 있는 고정요소는 트랜잭션 수행 전과 수행 완료 후의 상태가 같아야 한다.
- Isolation(독립성)
    - 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우 어떤 하나의 트랜잭션이라도 다른 트랜잭션의 연산에 끼어들수 없다.
    - 수행 중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조 할 수 없다.
- Durability(지속성)
    - 트랜잭션이 성공적으로 완료되었을 경우, 결과는 영구적으로 반영되어야 한다. -> commit -> 실행 결과를 DB반영

**. TCL(Transaction Controller Language): 트랜잭션 제어어
** 데이터의 일관성, 동시발생을 보장
COMMIT
ROLLBACK
SAVEPOINT -> 취소 지점

트랜잭션(Transaction): 작업 하나의 단위
COMMIT-> 작업 확정
ROLLBACK-> 작업 취소

** 트랜잭션의 상태 -> 시작 -> 완료
1. 활동 상태 : 트랜잭션이 수행을 시작하여 현재 실행되는 상태
2. 부분 완료 상태 : 트랜잭션에서 마지막 명령이 실행된 직후의 상태 즉, 트랜잭션의 모든 연산이 처리된 상태 -> commit직전
3. 실패 상태 : 오류 인해 장애가 발생, 트랜잭션의 수행이 중단된 상태
4. 완료 상태 : 트랜잭션이 성공적으로 완료된 상태를 의미, 최종결과를 데이터베이스에 반영 -> commit 직후
5. 철회 상태 : 트랜잭션 수행이 실패하여 Rollback연산을 실행한 상태, 수행된 트랜잭션 연산을 모두 취소하

고 트랜잭션이
수행되기 전의 상태로 복귀

*** 자동 commit
1. DDL -> create, drop, alter
2. DCL -> grant, revoke

***commit *** commit 수행 후에는 rollback이 불가능 하다.
DML -> insert, update ,delete -> commit를 수행 해야 DB반영 된다.

```
drop table test0628;

create table test0628(
    name varchar2(100) not null,
    age number not null
);
desc test0628;
select * from test0628;

-- rollback
insert into test0628(name,age) values('m1',11); -- a
commit; -- a대한 commit

insert into test0628(name,age) values('m2',21); --b
commit; -- b대한 commit

insert into test0628(name,age) values('m3',31);
insert into test0628(name,age) values('m4',41);
insert into test0628(name,age) values('m5',51);
commit;

insert into test0628(name,age) values('m6',61);
commit;
-- A  savepoint -> 자동 설정된 savepoint

delete from test0628 where name='m1';
savepoint p1; -- savepoint 설정

delete from test0628 where name='m2';
savepoint p2;

-- rollback
rollback;  -- commit 전단계로 이동
```

```
-- 특정 savepoint로 rollback
rollback to p1;

-- 특정 savepoint로 rollback
rollback to p2;

commit;

select * from test0628;
```

집합연산
1. 두테이블연산에집한연산자를사용하는방식
2. 여러질의결과를연결하여하나로결합하는방식을사용
3. UNION : 합집합(중복행제거)
4. UNION ALL : 합집합(중복행제거하지않음)
5. INTERSECTION : 교집합(중복행제거)
6. EXCEPT(MINUS-> 오라클) : 차집합(중복행제거)

2. 집합 함수: 테이블의 전체 행을 하나 이상의 컬럼을 기준으로 그룹화 하여 그 그룹 별 통계값을 출력하는 함수
- SUM : 그룹의 합계
- AVG : 그룹의 평균
- COUNT : 그룹의 개수
- MAX : 그룹의 최대값
- MIN : 그룹의 최소값
- STDEV : 그룹의 표준편차 -> 오라클 STDDEV_POP
- VAR : 그룹의 분산 -> 오라클 VAR_POP

**** COUNT(*) -> 전체 레코드 수
- 테이블에서 조건을 만족하는 행의 개수를 반환하는 함수 COUNT(속성명)
- 속성값이 NULL아닌 속성값의 개수 COUNT(DISTINCT(중복제거) 속성명)
- 속성값이 NULL이 아니며 중복되지 않는 속성값들의 개수

GROUP BY와 HAVING
GROUP BY- 특정 속성을 기준으로 테이블 전체를 그룹으로 설정
**** SELECT 절에는 집단 연산자나 GROUP BY에 사용한 속성명 만을 사용
```
select sum(value) from 테이블명 group by value;
```
**** 공통되는 속성값으로 그룹핑을 했으므로, 각 그룹에서 개별 튜플을 접근할 수 없다.
```
select sum(value) , id from 테이블명 group by value; --id오류
```
HAVING
- 각 그룹에 대한 제약 조건을 기술

- HAVING 절은 GROUP BY 절과 같이 사용
- WHERE 절은 전체 테이블의 조건 설정

분산(VAR)
- 각 값이 평균과 얼마나 떨어져 있는지에 대한 통계값
- 각 값과 평균의 차에 대한 차(즉, 편차)의 제곱의 평균
(sumi=1,n (xi- M)2)/N
표준 편차(STDEV)- 분산의 경우 편차에 대한 제곱으로 나타냄으로 평균과의 단위가 맞지 않음 - 단위를 맞추기 위하여 분산의 제곱근을 표준편차로 사용함
STDEV = VAR1/2

2. 집단 연산자
집단 함수
COUNT(*)
- 테이블에서 조건을 만족하는 행의 개수를 반환하는 함수 COUNT(속성명)
- 속성값이 NULL아닌 속성값의 개수 COUNT(DISTINCT 속성명)
- 속성값이 NULL이 아니며 중복되지 않는 속성값들의 개수
GROUP BY와 HAVING
GROUP BY- 특정 속성을 기준으로 테이블 전체를 그룹으로 나누기 위한 절
- SELECT 절에는 집단 연산자나 GROUP BY에 사용한 속성명 만을 사용할 수 있음 - 공통되는 속성값으로 그룹핑을 했으므로, 각 그룹에서 개별 튜플을 접근할 수 없음
HAVING- 각 그룹에 대한 제약 조건을 기술할 때 사용함- HAVING 절은 GROUP BY 절의 종속절임- WHERE 절은 테이블 전체에 대한 제약 조건을 나타냄

2. 집단 연산자
ROLLUP과 CUBE
다중 속성 GROUP BY
- 하나 이상의 속성들을 이용하여 그룹을 나누고, 그룹별로 다시 서브 그룹을 나누고자 할 때
GROUP BY 컬럼1, 컬럼2, ..., 컬럼n
ROLLUP 연산자
- GROUP BY 절의 그룹 조건에 따라서 그룹핑 하고 각 그룹에 대한 부분합을 구하는 연산자
- GROUB BY절에 n개의 속성 명이 있으면, n+1개의 그룹핑 조합이 나옴
CUBE 연산자
- GROUP BY 절의 그룹 조건에 따라서 그룹핑하고 각 그룹의 조합에 따른 부분합을 구하는 연산자
- GROUB BY 절에 n 개의 속성명이 있으면 2n개의 그룹핑 조합이 나옴
GROUPING SETS 함수
- 경우에 따라서 여러 개의 GROUP조건을 표시하고 싶은 경우, 부서,직급별 합을 보고 싶지 않은 경우 GROUPING SETS 함수를 이용함

1. 순위 함수
RANK 함수
RANK함수 over (order by 속성명 [asc|desc])
RANK() 함수
SELECT 속성명, RANK () OVER (ORDER BY 속성명 [asc|desc])
- 동률에 대하여 동일 등수 배정 - 비연속식 등수 배정

DENSE_RANK() 함수
SELECT 속성명, DENSE_RANK () OVER (ORDER BY 속성명)
- 동률에 대하여 동일 등수 배정 - 연속식 등수 배정
ROW_NUMBER() 함수
SELECT 속성명, ROW_NUMBER () OVER (ORDER BY 속성명)
- 동률에 대하여 임의 등수 배정 - 연속식 등수 배정
NTILE(n) 함수- 전체 튜플을 num개로 균등 분할하여 순위 지정
2. 그룹 별 순위
그룹 별 순위 지정
기존 RANK함수 문법- 전체 결과에 대한 속성값 기준 등수 지정이 됨
PARTITION BY 속성명
- 튜플들을 속성값에 따라서 그룹핑함
- 각 그룹에 대하여 순위 함수를 적용함
RANK() over (PARTITION BY dno ORDER BY salary desc)
그룹별 특정 등수의 정보를 보고 싶은 경우 - WHERE 절을 같이 활용함
그룹 별 집단 함수
그룹 별 집단 함수의 적용

--- 인덱스의 개념 (색인)
-인덱스(Index)
1) 검색 성능을 향상 시키기 위한 부가적인 자료 구조(빠른 검색) ***
2) 질의 명령문의 검색 속도를 향상시키기 위해 칼럼에 대해 생성하는 객체
3) 포인터(주소값)를 이용하여 테이블에 저장된 데이터에 접근

-. 인덱스 생성 / 삭제 구문
CREATE INDEX 색인명 ON 테이블명(속성명, 속성명,...)
DROP INDEX 색인명 ON 테이블명

create index 인덱스명 on 테이블(칼럼명1,칼럼명2);
create index 인덱스명 on 테이블(칼럼명1);
create UNIQUE index 인덱스명 on 테이블(칼럼); --칼럼은 반드시 UNIQUE

drop INDEX 인덱스명;

** 기본키
1 테이블이 기본키에 대해서는 자동으로 고유색인이 생성됨
⇨ Primary Index
• 기본키는 중복을 허용하지 않음
2 새로운튜플을삽입할때마다키값이고유값인지검사해야함
3 테이블에속한튜플들이많다면매우느림
인덱스 성질 중에서 기본키가 설정 된 칼럼은 인덱스를 설정 할수 있다.

SELECT * FROM USER_INDEXES WHERE TABLE_NAME = '테이블명'
SELECT * FROM ALL_IND_COLUMNS WHERE TABLE_NAME ='테이블명'

2. 뷰(View)의 개념
1) 뷰의 개념 -> 기존테이블에서 필요한 데이터만 별도로 테이블화 시킨 가상 객체
1. VIEW란 기존의 존재하는 테이블로 만들어진 가상의 테이블****
2. VIEW를 통해 필요한 데이터만 정의해서 관리 ***
3. VIEW에 정의되지 않은 데이터들은 사용자에게 보이지 않는다
4. 데이터를 비교적 안전하게 관리
5. 추가적으로 VIEW를 통해 쿼리를 간단히 할 수 있다

** 장점 -> 기존 테이블과 독립적이다. 관리하기 쉽다. 보안이용(기존테이블변경X)
** 단점 -> 뷰정의 변경 불가,

View생성

```
 --뷰생성1                                                --기본 테이블1
CREATE OR REPLACE VIEW  뷰이름
AS
SELECT 칼럼1,칼럼2  FROM  테이블 ;
 --뷰생성2
CREATE  VIEW  뷰이름
AS
SELECT 칼럼1,칼럼2  FROM  테이블 ;
```

```
create or replace view member_02_view as

    select m1.custno, m1.grade, m1.city from member_07 m1;

select * from member_02_view;
```

View별칭 사용

```
CREATE OR REPLACE VIEW 뷰이름( a1, a2 )  AS
    SELECT 칼럼1,칼럼2        FROM 테이블 ;
```

View수정

```
CREATE OR REPLACE VIEW 뷰이름( a1, a2 )  AS
    SELECT 칼럼1,칼럼2        FROM 테이블
        where 칼럼='값';
```

View삭제

```
DROP VIEW 뷰이름;
```

View 구조

```
desc View이름;
```

--View 조회
SELECT * FROM USER_VIEWS;

뷰의 생성 구문
뷰의 삭제 구문
DROP VIEW 뷰이름
뷰의 종류- 단순 뷰 : 하나의 기본 테이블 위에 정의된 뷰- 복합 뷰 : 두 개 이상의 기본 테이블로부터 파생된 뷰
뷰에 대한 갱신 연산
- 무결성 제약 조건, 표현식, 집단연산, GROUP BY 절의 유무에 따라서 DML문의 사용이 제한적임
인라인 뷰
인라인 뷰란?
- 하나의 질의문 내에서만 생성되어 사용 되어지고 질의문 수행 종료 후에는 사라지는 뷰
- 뷰의 명시적인 선언(즉, Create View 문)이 없음
- FROM 절에서 참조하는 테이블의 크기가 클 경우, 필요한 행과 속성만으로 구성된 집합으로 정의하여 질의문을 효율적으로 구성함
- FROM절에서 서브 쿼리를 사용하여 생성하는 임시 뷰 WITH 절 - 인라인 뷰의 또 다른 정의 방법 - FROM 절에 임시 질의 결과를 정의하는 대신 WITH 절을 이용하여 임시 테이블을 생성함 뷰의 정의 보기- 뷰의 정의 내용을 보고 싶을 경우 SP_HELPTEXT라는 저장 프로시져를 이용함 - 저장 프로시져를 수행하는 명령문 : EXEC

*** 트랜잭션(Transaction)이란? -> spring boot JPA @Transanctional
- 트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 수행 되어야할 일련의 연산들
- 트랜잭션은 작업의 완전성을 보장 , 작업 단위. -> 현금 인출기
- 트랜잭션은 SELECT, UPDATE, INSERT, DELETE와 같은 연산을 수행하여 데이터베이스의 상태를 변화시키는 작업의 단위다.

***트랜잭션의 특징(ACID)
- Atomicity(원자성)
  - 트랜잭션이 데이터베이스에 모두 반영되던지, 아니면 전혀 반영 되지 않아야 한다.
  - 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 한다.
- Consistency(일관성)
  - 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다.
  - 시스템이 가지고 있는 고정요소는 트랜잭션 수행 전과 수행 완료 후의 상태가 같아야 한다.
- Isolation(독립성)
  - 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우 어떤 하나의 트랜잭션이라도 다른 트랜잭션의 연산에 끼어들수 없다.
  - 수행 중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조 할 수 없다.
- Durability(지속성)
  - 트랜잭션이 성공적으로 완료되었을 경우, 결과는 영구적으로 반영되어야 한다. -> commit -> 실행 결과를 DB반영

**. TCL(Transaction Controller Language): 트랜잭션 제어어
** 데이터의 일관성, 동시발생을 보장
COMMIT
ROLLBACK
SAVEPOINT -> 취소 지점

트랜잭션(Transaction): 작업 하나의 단위
COMMIT-> 작업 확정
ROLLBACK-> 작업 취소

** 트랜잭션의 상태 -> 시작 -> 완료
1. 활동 상태 : 트랜잭션이 수행을 시작하여 현재 실행되는 상태
2. 부분 완료 상태 : 트랜잭션에서 마지막 명령이 실행된 직후의 상태 즉, 트랜잭션의 모든 연산이 처리된 상태 -> commit직전
3. 실패 상태 : 오류 인해 장애가 발생, 트랜잭션의 수행이 중단된 상태
4. 완료 상태 : 트랜잭션이 성공적으로 완료된 상태를 의미, 최종결과를 데이터베이스에 반영 -> commit 직후
5. 철회 상태 : 트랜잭션 수행이 실패하여 Rollback연산을 실행한 상태, 수행된 트랜잭션 연산을 모두 취소하고 트랜잭션이
수행되기 전의 상태로 복귀

*** 자동 commit
1. DDL -> create, drop, alter
2. DCL -> grant, revoke

***commit *** commit 수행 후에는 rollback이 불가능 하다.
DML -> insert, update ,delete -> commit를 수행 해야 DB반영 된다.

```
drop table test0628;

create table test0628(
    name varchar2(100) not null,
    age number not null
);
desc test0628;
select * from test0628;

-- rollback
insert into test0628(name,age) values('m1',11); -- a
commit; -- a대한 commit

insert into test0628(name,age) values('m2',21); --b
commit; -- b대한 commit

insert into test0628(name,age) values('m3',31);
```

```
insert into test0628(name,age) values('m4',41);
insert into test0628(name,age) values('m5',51);
commit;

insert into test0628(name,age) values('m6',61);
commit;
-- A  savepoint -> 자동 설정된 savepoint

delete from test0628 where name='m1';
savepoint p1; -- savepoint 설정

delete from test0628 where name='m2';
savepoint p2;

-- rollback
rollback;  -- commit 전단계로 이동

-- 특정 savepoint로 rollback
rollback to p1;

-- 특정 savepoint로 rollback
rollback to p2;

commit;

select * from test0628;
```