DBMS 종류는 굉장히 많다

26페이지 정리 필요, 61페이지와 유사한 표 만드는 것이 시험 문제  
값을 insert 하는 문제(컬럼명 생략하려면 values에 전부 넣어줘야 함.)

디비 구성 파악에 중요한 다이어그램 그리는 것도 시험 문제

revoke 권한을 뺐음

**1. CHAR**

CHAR 데이터 타입은 고정된 문자열을 저장하는 데 사용합니다.

테이블을 만들 때, 무조건 고정된 길이를 지정해줘야 하고 범위는 1~2000 bytes입니다.

한글의 경우 글자당 2byte이므로 1000글자 저장 가능. 영문, 숫자의 경우 1byte입니다.

그러나 오라클에서는 캐릭터 셋에 따라서 한 글자를 3바이트로 인식하는 경우도 있다고 하니 주의!

첫째, CHAR로 데이터 타입을 선언하게 되면 고정된 길이를 가집니다.

둘째, CHAR를 데이터 타입으로 선언한 변수에 고정된 길이보다 작은 길이의 문자를 넣어도 남은 부분을 공백으로 다 채웁니다.

셋째, 만약 더 큰 길이를 입력하게 되면 오라클 DB는 에러를 반환합니다.

즉, 2000바이트로 선언했으면 무조건 2000바이트를 가지게 되는 것입니다.

**2. VARCHAR, VARCHAR2**

VARCHAR, VARCHAR2는 가변 길이로 문자열을 저장할 수 있는 데이터 타입입니다. 만약 우리가 VARCHAR2로 변수를 만들면 1~4000byte까지의 길이를 지정해줄 수 있습니다.

char와의 차이점이라면 우리가 선언한 문자열보다 더 짧은 문자열을 선언하게 될 경우 char의 경우 공백으로 남은 부분을 메꾸지만 varchar와 varchar2는 그렇지 않다는 점입니다.

그렇기 때문에, varchar2를 사용하면 메모리 상의 이점을 얻게 되어 더 많이 사용됩니다.

**3. VARCHAR와 VARCHAR2 차이점**

둘의 차이점은 없다. 실제로 오라클 공식 문서에는 두 데이터 타입이 synoymous, 즉 동의어라고 나와 있다.

그러나 varchar2를 사용하는 것을 추천. 왜냐하면 varchar는 나중에 다른 방식의 데이터 타입으로 변경될 수 있기 때문.

프로그램을 만들고 운영 중인데 나중에 varchar로 선언한 것을 다 varchar2로 바꿔야 할 수도 있다.

insert into //(//) values //(//);

select \* from //;

정말 긴 길이로 입력할 때

binary -> BLOB (8 to 128TB)

char -> CLOB (8 to 128TB)

**primary key = 유일성**

-not null 내장되어 있음

-다른 항목과 절대로 중복되어 나타날 수 없는 단일 값(unique)

-절대 null(아무런 값이 없는 상태) 값을 가질 수 없다

-하나 이상의 컬럼이 그룹화 되어 기본 키로도 쓰일 수도 있다

-기본 키를 하나까지만 가질 수 있다.

-관계형 DB 이론상 모든 테이블은 반드시 하나의 기본 키를 가져야 한다

-코드 작성으로는 not null 입력을 안해도 되지만 편의 기능으로 지정 시에는 반드시 체크

\*어떤 column을 null불가에서 null 가능으로 바꾸려면 만들어놓은 해당 column 중에 null이 있으면 안 된다

**\*rollback** : 커밋 이전의 데이터를 복구시켜준다

-TRANSACTION(INSERT, UPDATE, DELETE)작업 내용을 취소

**\*commit** : 변경된 내용을 모두 영구 저장한다.

-모든 작업을 정상적으로 처리하겠다고 확정하는 명령어

-하나의 트랜젝션 과정을 종료하게 된다.

-TRANSACTION(INSERT, UPDATE, DELETE)작업 내용을 실제 DB에 저장

-이전 데이터가 완전히 UPDATE된다.

-모든 사용자가 변경한 데이터의 결과를 볼 수 있다.

\*rollback, commit 명령어의 장점 : 데이터 무결성 보장, 변경사항 확인 가능, 논리적으로 작업을 그룹화

\*num number(2, 3) (정밀도, 스케일) -> 2가 전체 자리수가 아니다

--소수점 밑에서 첫 1이상의 숫자가 나오고 2자리까지 판단 후 버림

--소수점 4번째에서 반올림 후 3번째 자리까지 입력

\***timestamp**를 활용하면 커밋 이후에도 데이터를 복구할 수 있다

TIMESTAMP DATETIME

타입 숫자형 문자형

용량 4byte 8byte

입력 Default Insert (AUTO) 명시적 INSERT

형식 1970~01-01 ~ 2038-01-19 (UTC) 1000-00-00 ~ 9999-99-99

\***unique**로 인한 무결성 제약 조건 위배

-유일성, 중복 불허

-null 허용

-null 값 가능 여부에 따라 기본 키와 구분

-유니크에 기본키가 속한 느낌

\***외래키**

-참조 키

-다른 테이블의 기본 키를 참조하는 것

-table DEPT, table EMP

부모가 DEPT 자식이 EMP, 따라서 부모에 pk(DEPTNO)가 설정이 안 되어있으면 행 추가 불가

\***슈퍼키** : 튜플을 유일하게 식별할 수 있는 하나의 속성 또는 속성의 집합

-튜플을 유일하게 식별할 수 있는 값이면 모두 슈퍼키가 될 수 있다

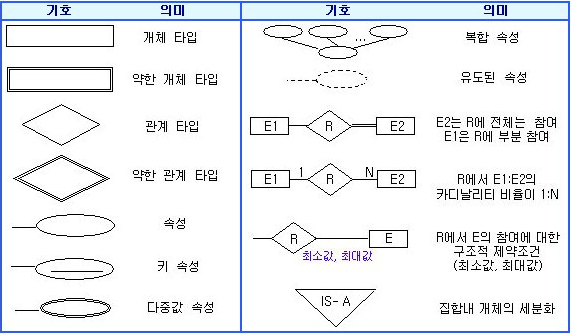
\***후보키 :** 튜플을 유일하게 식별할 수 있는 속성의 최소 집합

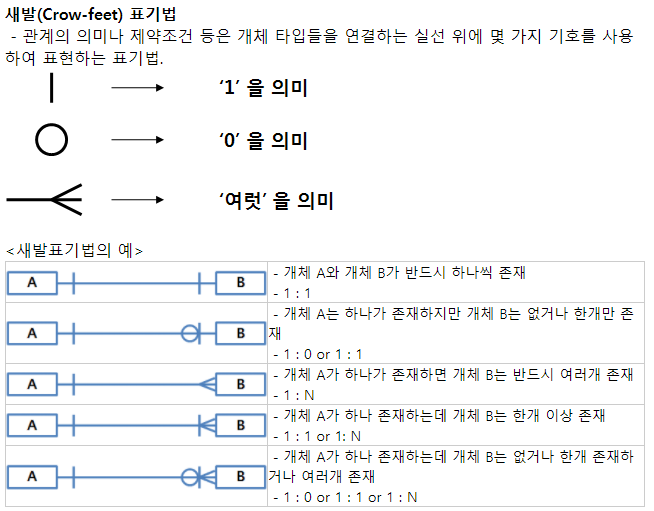
-모든 릴레이션은 반드시 하나 이상의 후보키를 가져야 한다.

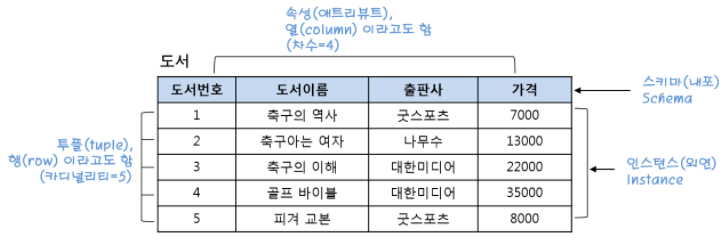
\***대체키** : 기본키로 선정되지 않은 후보키

-후보키가 두 개 이상일 때 기본키를 제외한 나머지 후보키(보조키)

**E-R 관계도(개체 관계도)**







**릴레이션** : 행과 열로 구성된 테이블

**속성** : 릴레이션(도서) 스키마 열

-고유한 이름

-릴레이션이 다르면 같은 속성 상관X

-단일 값

-한 속성의 값은 모두 같은 도메인 값을 가진다(같은 열은 그 속성에 대한 도메인 값만 가진다)

-속성과 튜플 순서는 상관없다

-릴레이션 내의 중복 튜플은 불허

**튜플** : 행

**차수(degree)** : 속성의 개수, 유효한 릴레이션의 최소 차수 = 1

**도메인** : 속성이 가질 수 있는 값의 집합

**스키마** : 릴레이션이 어떻게 구성되는지, 어떤 정보를 담고 있는지 기본적인 구조를 정의

-첫 행 헤더에 나타내며 각 데이터의 특징을 나타내는 속성, 자료 타입등의 정보를 담음

**인스턴스** : 정의된 스키마에 따라 테이블에 실제 저장된 데이터의 집합

**카디날리티** : 튜플 수

**데이터 모델링 : 시스템 개발에 있어서 가장 핵심적인 과정**

데이터 모델은 안정성, 유연성, 성능 등에 영향을 미친다.

**데이터의 무결성**을 해결해준다. => 데이터의 **정확성, 일관성, 유효성**이 유지되는 것

-완전한 수명 주기를 거치며 데이터의 정확성과 일관성을 유지하고 보증하는 것

-개체 무결성 : 모든 테이블이 기본 키(primary key)를 가져야 하며 기본 키로 선택된 열은 고유하여야 하며 빈 값은 허용치 않음을 규정

-참조 무결성 : 모든 외래 키 값은 두 가지 상태 가운데 하나에만 속함을 규정

-범위 무결성 : 정의된 범위에서 관계형 데이터베이스의 모든 열이 선언되도록 규정

-도메인 무결성 : 같은 속성에 사용되느니 값들은 같은 성격의 값. 기본 값이나 널 여부, 체크 조건 등으로 지켜질 수 있음

파급 효과(LEVERAGE) : 데이터 설계를 잘하여 변경 작업에 대한 위험(파급 효과)을 방지해야한다

복잡한 정보 요구 사항의 간결한 표현(CONCISENESS)

데이터 품질(DATA QUALITY)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **DELETE** | **TRUNCATE** | **DROP** |
| **명령어 종류** | DML | DDL | DDL |
| **처리속도** | 느림 | 빠름 | 빠름 |
| **Commit** | 사용자가 직접 | 자동 | 자동 |
| **Rollback 가능 여부** | Commit 이전 가능 | 불가 | 불가 |
| **삭제되는 정도** | 데이터만 삭제 | 테이블을 create 상태로 되돌림 | 테이블까지 완전히 제거 |

**TABLE, VIEW 차이**

테이블은 실질적인 데이터가 있지만, 뷰는 데이터가 없고 SQL만 저장한다.

뷰를 생성하기 위해 테이블이 존재해야 한다.

뷰를 사용하는 이유

-뷰를 통해 데이터에 접근하면 뷰에 없는 데이터를 보호할 수 있다.

-복잡한 쿼리를 단순화할 수 있다.

뷰의 단점은 내용 삽입, 삭제, 갱신, 연산에 제약

단순 뷰는 가능하지만 복합 뷰는 불가능하다