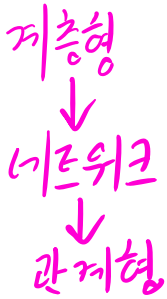
- 데이터베이스의 종류

1. 데이터베이스는 데이터를 어떠한 형태의 자료구조로 사용하느냐에 따라서 나누어진다.
2. 데이터베이스의 종류는 계층형 데이터베이스, 네트워크형 데이터베이스, 관계형 데이터베이스 등이 있다.
3. **계층형 데이터베이스**는 트리 형태의 자료구조에 데이터를 저장하고 관리하며, **네트워크 데이터베이스**는 오너와 멤버 형태로 데이터를 저장한다.
4. **계층형 데이터베이스**는 1대 N 관계를 표현한다.
5. **네트워크 데이터베이스**는 1대 N 관계 뿐만 아니라, M대 N 표현도 가능하다. 즉, 계층형 데이터베이스를 개선한 것이다.
6. 관계형 데이터베이스는 릴레이션에 데이터를 저장하고 관리한다.
7. 관계형 데이터베이스는 릴레이션을 사용해서 **집합 연산**과 **관계 연산**을 할 수 있다.
8. 집합 연산 : 합집합, 차집합, 교집합, 곱집합(Cartesian product)
9. 관계 연산 : 선택 연산(SELECT), 투영 연산, 결합 연산(JOIN), 나누기 연산



- 데이터베이스와 데이터베이스 관리 시스템의 차이점

1. 데이터베이스 관리 시스템(**DBMS**)은 계층형 데이터베이스, 네트워크 데이터베이스, 관계형 데이터베이스 등을 관리하기 위한 소프트웨어를 의미한다.
2. DBMS의 종류에는 Oracle, MS-SQL, MySQL, Sybase 등이 있으며 모두 관계형 데이터베이스를 지원한다.

- 테이블의 구조

1. 관계형 데이터베이스는 릴레이션에 데이터를 저장하고 릴레이션을 사용해서 집합 연산 및 관계 연산을 지원하여 다양한 형태로 데이터를 조회할 수 있다.
2. 릴레이션은 최종적으로 데이터베이스 관리 시스템에서 테이블로 만들어진다.
3. 기본키는 하나의 테이블에서 **유일성**과 **최소성**, **Not Null**을 만족하면서 해당 테이블을 대표하는 것(**대표성**)이다.
4. 테이블은 행과 컬럼으로 구성된다. 그중에서 행은 하나의 테이블에 저장되는 값으로 '튜플', '인스턴스'라고도 한다.
5. 컬럼은 어떤 데이터를 저장하기 위한 필드로 속성(Attribute)이라고도 한다. . 컬럼 = 필드 = 속성
6. 외래키는 다른 테이블의 기본키를 참조하는 칼럼이다.
7. **외래키는 관계 연산 중에서 결합 연산(JOIN)을 하기 위해서 사용한다.**

- SQL 종류

1. SQL은 데이터 정의, 데이터 조작, 데이터 제어 등의 기능을 지원한다.
2. DDL : 관계형 데이터베이스의 구조를 정의하는 언어이다. 즉, 데이터베이스 테이블을 생성하거나 변경, 삭제하는 것으로 데이터를 저장할 구조를 정의하는 언어이다. CREATE, ALTER, DROP, RENAME문이 있다.
3. DML : 테이블에서 데이터를 입력, 수정, 삭제, 조회한다. 즉, 데이터 구조가 DDL로 정의되면 해당 데이터 구조에 데이터를 입력하거나 수정, 삭제, 조회 할 수 있다. INSERT, UPDATE, DELETE, SELECT문이 있다.
4. DCL(Data Control Language) : 데이터베이스 사용자에게 권한을 부여하거나 회수한다. 즉, DDL로 정의된 구조에 어떤 사용자가 접근할 수 있는지 권한을 부여하는 것이다. (GRANT, REVOKE)
5. TCL(Transaction Control Language) : 트랜잭션(데이터베이스의 작업을 처리하는 단위)을 제어하는 명령이다. (COMMIT, ROLLBACK, SAVEPOINT)
6. 작업의 순서를 보면 데이터베이스의 사용자에게 권한을 부여하고(DCL), 권한이 부여되면 DDL로 데이터 구조를 정의한다. 데이터 구조가 정의되면 데이터를 입력한 후에 개발자 및 사용자가 그 데이터를 조회하는 것이다. (DML)**[DCL -> DDL -> DML]**

- 트랜잭션

1. 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.
2. 특성 : **원자성**(트랜잭션은 데이터베이스 연산의 전부가 실행되거나 전혀 실행되지 않아야 한다. 즉, 트랜잭션의 처리가 완전히 끝나지 않았을 경우는 실행되지 않은 상태와 같아야 한다.), **일관성**(트랜잭션 실행 후에도 일관성이 유지되어야 한다.), **고립성**(트랜잭션 실행 중에 생성하는 연산의 중간결과는 다른 트랜잭션이 접근할 수 없다. 즉, 부분적인 실행 결과를 다른 트랜잭션이 볼 수 없다.), **영속성**(트랜잭션이 그 실행을 성공적으로 완료하면 그 결과는 영구적 보장이 되어야 한다.)

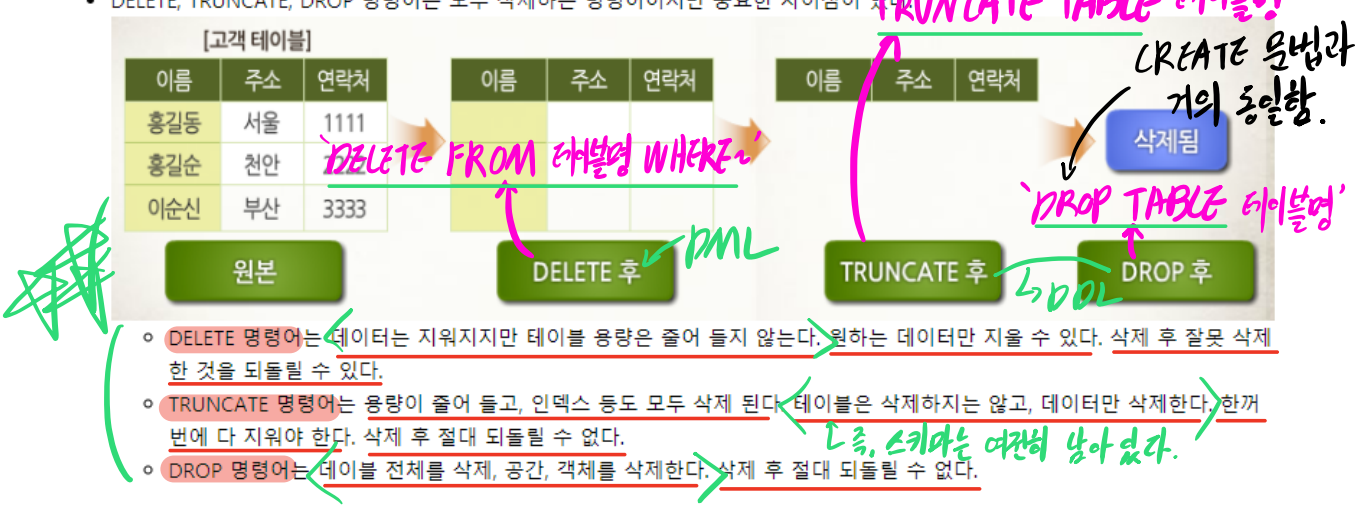
- SQL문의 실행 순서

1. SQL 실행 순서 : **파싱 -> 실행 -> 인출**
2. 파싱 : SQL 문법 구문 분석 후, 분석된 결과가 Library Cache에 저장
3. 실행 : 옵티마이저가 수립한 실행 계획에 따라 분석된 SQL을 실행
4. 인출 : 결과물을 사용자에게 전송

- DELETE, TRUNCATE, DROP 명령어의 차이점

6) DELETE, TRUNCATE, DROP 명령어의 차이점

- DELETE, TRUNCATE, DROP 명령어는 모두 삭제하는 명령어이지만 중요한 차이점이 있다.



- DELETE 명령어는 데이터는 지워지지만 테이블 용량은 줄어 들지 않는다. 원하는 데이터만 지울 수 있다. 삭제 후 잘못 삭제한 것을 되돌릴 수 있다.
- TRUNCATE 명령어는 용량이 줄어 들고, 인덱스 등도 모두 삭제 된다. 테이블은 삭제하지는 않고, 데이터만 삭제한다. 한꺼번에 다 지워야 한다. 삭제 후 절대 되돌릴 수 없다.
- DROP 명령어는 테이블 전체를 삭제, 공간, 객체를 삭제한다. 삭제 후 절대 되돌릴 수 없다.

- 제약조건 활용하여 외래키 설정 주의사항 : 'CONSTRAINTS 제약조건명 FOREIGN KEY(칼럼명) **REFERENCES 참조테이블(컬럼명)**'에서 'REFERENCES'에 'S'가 붙는 것이 핵심

- 제약조건 활용하여 기본키 설정 방법 : 'CONSTRAINTS 제약조건명 PRIMARY KEY(컬럼명)'

- 테이블 명 변경 : 'ALTER TABLE ~ RENAME TO ~'

- 테이블 내 컬럼명 변경 : 'ALTER TABLE ~ RENAME COLUMN ~ TO ~'

- 테이블 내 데이터 값 수정(UPDATE는 DML임) : 'UPDATE 테이블명 **SET** ~ WHERE ~'

- 내장 함수는 보통 SELECT문에 작성된다.

- NULL값 조회 : ‘WHERE 컬럼명 IS NULL’
- NULL이 아닌 값 조회 : ‘WHERE 컬럼명 IS NOT NULL’