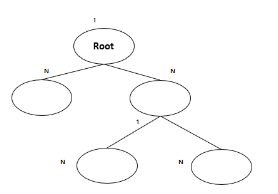
관계형 데이터베이스

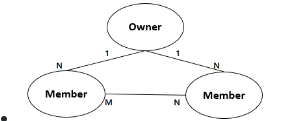
* 관계형 데이터베이스는 릴레이션과 릴레이션의 조인 연산을 통해 합집합, 교집합, 차집합 등을 만들 수 있다.

데이터베이스와 데이터베이스 관리 시스템의 차이점

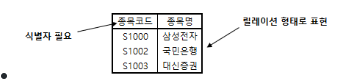
* 데이터베이스의 종류
* 계층형
  + 트리 형태의 자료구조에 데이터를 저장하고 관리
  + 1대N 관계 표현



* 네트워크 형
  + 오너(Owner)와 멤버(Member) 형태로 데이터 저장
  + 1대N과 함께 M대N 표현도 가능



* 관계형
  + 릴레이션에 데이터를 저장하고 관리, 릴레이션을 사용하여 집합연산, 관계연산 가능



* 데이터베이스 관리 시스템(DBMS)
* 계층형/네트워크/관계형 데이터베이스 등을 관리하기 위한 소프트웨어

관계형 데이터베이스 집합 연산과 관계 연산

* 관계형 데이터베이스의 특징
* 릴레이션을 사용해서 집합 연산과 관계 연산을 할 수 있다.

집합 연산

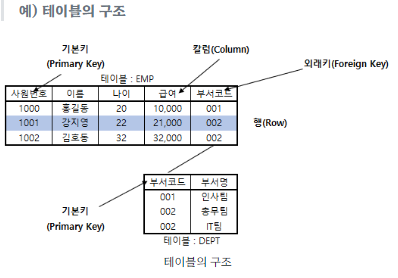
* 합집합(Union)
  + 2개의 릴레이션을 하나로 합하는 것
  + 중복된 행(튜플)은 한 번만 조회
* 차집합(Difference)
  + 본래 릴레이션에는 존재하고 다른 릴레이션에는 존재하지 않는 것을 조회
* 교집합(Intersection)
  + 2개의 릴레이션 간에 공통된 것을 조회
* 곱집합(Cartesian Product)
  + 각 릴레이션에 존재하는 모든 데이터를 조합하여 연산

관계 연산

* 선택 연산(Selection)
  + 릴레이션에서 조건에 맞는 행(튜플)만 조회
* 투명 연산(Projection)
  + 릴레이션에서 조건에 맞는 속성만 조회
* 결합 연산(Join)
  + 여러 릴레이션의 공통된 속성을 사용해서 새로운 릴레이션을 만들어냄
* 나누기 연산(Division)
  + 기준 릴레이션에서 나누는 릴레이션이 가지고 있는 속성과 동일한 값을 가지는 행(튜플)을 추출하고, 나누는 릴레이션의 속성을 삭제한 후 중복된 행을 제거하는 연산

테이블의 구조

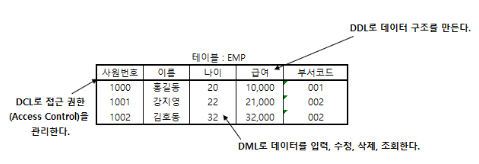
* 관계형 데이터베이스는 릴레이션에 데이터를 저장하고, 릴레이션을 사용해서 집합 연산 및 관계 연산을 지원하여 다양한 형태로 데이터를 조회할 수 있다.
* 릴레이션은 최종으로 데이터베이스 관리 시스템에서 테이블로 만들어진다.



* 기본키(Primary Key): 유일성, 최소성, Not Null 해당 테이블을 대표 (사원번호)
* 행(Row)과 칼럼(Column)
  + 행: 하나의 테이블에 저장되는 값으로 튜플이라고 함
  + 칼럼: 어떤 데이터를 저장하기 위한 필드로 속성이라고 함
* 외래키(Foreign Key): 다른 테이블의 기본키를 참조(조인)하는 칼럼
  + EMP테이블의 부서코드
  + 결합 연산(Join)을 하기위해 사용함

SQL종류

* DDL(Data Definition Language)
  + 관계형 데이터베이스의 구조를 정의하는 명령어
  + 데이터베이스 테이블을 생성하거나 변경, 삭제
  + Ex. CREATE, ALTER, DROP, RENAME 등
* DML(Data Manipulation Language)
  + DDL로 정의된 데이터 테이블에서 데이터를 입력, 수정, 삭제, 조회하는 명령어
  + Ex. INSERT, UPDATE, DELETE, SELECT 등
* DCL(Data Control Language)
  + 데이터베이스 사용자에게 권한을 부여하거나 회수하는 명령어
  + DDL로 정의된 구조에 어떤 사용자가 접근할 수 있는지 권한 부여
  + Ex. GRANT, REVOKE, TRUNCATE 등
* TCL(Transaction Control Language)
  + 트랜잭션을 제어하는 명령어
  + Ex. COMMIT, ROLLBACK, SAVEPOINT 등



* 작업의 순서
  + 데이터베이스의 사용자에게 권한을 부여한다.
  + 권한이 부여되면 사용자는 DDL로 데이터 구조를 정의 한다.
  + 데이터 구조가 정의되면 사용자는 데이터를 입력한다.
  + 개발자 및 사용자가 입력한 데이터를 조회한다.
* 트랜잭션: 데이터베이스 작업을 처리하는 단위
* 트랜잭션 특성
* 원자성
  + 트랜잭션은 데이터베이스 연산의 전부가 실행되거나 전해 실행되지 않아야 함
  + 즉, 트랜잭션의 처리가 완전히 끝나지 않았을 경우 실행되지 않은 상태와 같아야 함
* 일관성
  + 트랜잭션 실행 결과로 데이터베이스의 상태가 모순되지 않아야 함
  + 트랜잭션 실행 후에도 일관성이 유지되어야 함
* 고립성
  + 트랜잭션 실행 중에 생성하는 연산의 중간 결과는 다른 트랜잭션이 접근 불가
  + 즉, 부분적인 실행 결과를 다른 트랜잭션이 볼 수 없다.
* 영속성
  + 트랜잭션이 그 실행을 성공적으로 완료하면 그 결과는 영구적 보장이 되어야 함

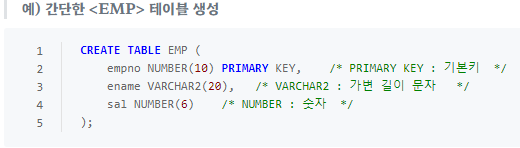
SQL문의 실행 순서

* 개발자가 작성한 SQL문(DDL, DML. DCL 등)은 3단계를 걸쳐서 실행된다.
* SQL문의 문법을 검사하고 구문분석을 한다.
* 구문분석 이후에 SQL을 실행한다.
* SQL이 실행되면 데이터를 인출하게 된다.
* 파싱(Parsing)
  + SQL문의 문법을 확인하고 구문분석 한다.
* 실행(Execution)
  + 옵티마이저(Optimizer)가 수립한 실행 계획에 따라 SQL을 실행한다.
* 인출(Fetch)
  + 데이터를 읽어서 전송한다.

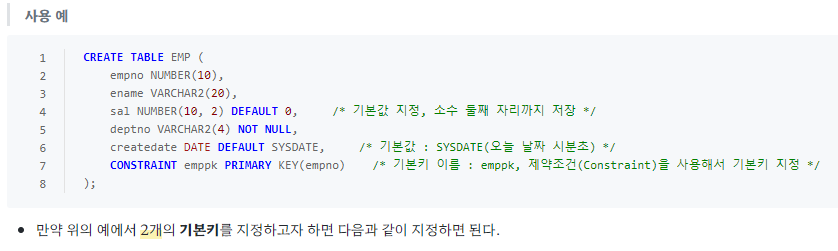
DDL (Data Definition Language)

테이블 생성

* CREATE TABLE
  + 새로운 테이블 생성
  + 생성할 때 기본키, 외래키, 제약사항 등 설정 가능
* ALTER TABLE
  + 생성된 테이블 변경
  + 칼럼을 추가하거나 변경, 삭제 가능
  + 기본키를 설정하거나 외래키 설정 가능
* DROP TABLE
  + 해당 테이블 삭제
  + 테이블의 데이터 구조뿐만 아니라 저장된 데이터도 모두 삭제됨
* 기본적인 테이블 생성



* 제약조건 사용 테이블 생성
  + 기본키, 외래키, 기본값, Not Null 등은 테이블을 생성할 때 지정 가능



* 외래키를 지정하려면 마스터 테이블이 생성되어 있어야 함



* 테이블 생성시 CASCADE 사용
  + 테이블을 생성할 때 CASCADE 옵션을 사용할 수 있다.
  + 참조관계가 있을 경우, 참조 데이터를 자동으로 반영할 수 있게 하는 옵션



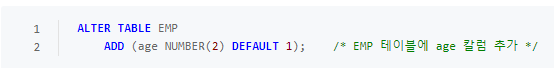
* ON DELETE CASCADE 옵션
* 자신이 참조하고 있는 테이블(<DEPT>)의 데이터가 삭제되면 자동으로 자신(<EMP>)도 삭제되는 옵션
* 이 옵션을 사용하면 참조 무결성을 준수 가능
  + <DEPT>에 해당 부서번호가 없는데 <EMP>에는 해당 부서번호가 있으면 위배

테이블 변경

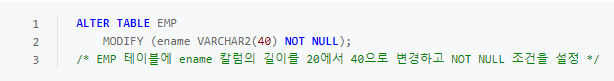
* 테이블명 변경
* ALTER TABLE ~ RENAME TO 구문



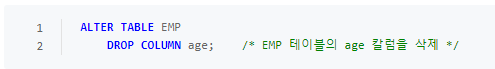
* 칼럼 추가
* ALTER TABLE ~ ADD 구문



* 칼럼 변경
* ALTER TABLE ~ MODIFY
* 칼럼 변경할 때 데이터 타입 변경, 데이터 길이 변경, 제약조건 설정 가능
* 데이터 타입 변경할 때 기존 데이터 타입과 다르게 변경하면 에러 발생



* 칼럼 삭제
* ALTER TABLE ~ DROP COLUMN 구문



* 칼럼명 변경
* ALTER TABLE ~ RENAME COLUMN ~ TO



테이블 삭제

* DROP TABLE 구문 => 테이블의 구조와 데이터를 모두 삭제



* CASCADE CONSTRAINT 옵션 사용
* 해당 테이블의 데이터를 외래키로 참조한 슬레이브 테이블과 관련된 제약사항도 삭제할 때 사용



뷰(View) 생성과 삭제

* 뷰(View): 테이블로부터 유도된 가상의 테이블
* 실제 데이터는 없고 테이블을 참조해서 원하는 칼럼만 조회할 수 있도록
* 뷰는 데이터 딕셔너리에 SQL문 형태로 저장하되, 실행 시에 참조

뷰의 특징

* 참조한 테이블이 변경되면 뷰도 변경된다.
* 뷰의 검색은 참조한 테이블과 동일하게 할 수 있지만, 뷰에 대한 입력, 수정, 삭제는 제약
* 특정 칼럼만 조회시켜서 보안성 향상
* 한번 생성한 뷰는 변경할 수 없고, 변경을 원하면 삭제 후 재생성 해야함
* ALTER문을 사용해 뷰 변경 불가

뷰의 장점과 단점

* 장점
* 특정 칼럼만 조회 할 수 있기 때문에 보안 기능
* 데이터 관리와 SELECT문이 간단해짐
* 하나의 테이블에 여러 개의 뷰 생성 가능
* 단점
* 뷰는 독자적 인덱스 만들 수 없음
* 삽입, 수정, 삭제 연산 제약/데이터 구조 변경 불가



DML(Data Manipulation Language)

INSERT

* 테이블에 데이터를 입력하는 DML문



* 데이터를 삽입 하려면 테이블명(칼럼명), 데이터순으로 입력



* 데이터를 입력할 때 문자열을 입력하는 경우 작은 따옴표 사용
* 특정 테이블의 모든 칼럼에 대한 데이터를 삽입하는 경우 칼럼명 생략 가능



* 주의사항
* INSERT문을 실행했다고 데이터 파일에 저장되는 것은 아니다
* 최종적으로 데이터를 저장하려면 TCL문인 COMMIT을 실행해야 함
* 만약 Auto Commit(SET AUTO COMMIT ON)으로 설정된 경우 COMMIT 안해도 됨

SELECT문으로 입력

* SELECT문을 사용하여 데이터를 조회해서 해당 테이블에 바로 삽입할 수 있다.
* 단, 입력되는 테이블은 사전에 생성되어 있어야 함



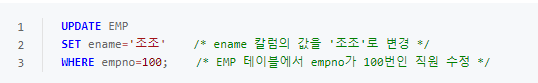
NOLOGGING 사용

* 데이터베이스에 데이터를 입력하면 로그 파일(Log File)에 그 정보를 기록한다.
* Check Point 라는 이벤트가 발생하면, 로그 파일의 데이터를 데이터 파일에 저장한다.
* NOLOGGING 옵션
  + 로그 파일의 기록을 최소화시켜서 입력 시 성능을 향상시키는 방법
  + 버퍼캐시 라는 메모리 영역을 생략하고 기록한다.



UPDATE

* UPDATE (테이블) SET (칼럼=’바꿀 값’) WHERE (조건)
* 원하는 조건으로 데이터를 검색해서 해당 데이터를 수정 가능
* 조건문(WHERE)을 입력하지 않으면 모든 데이터가 수정 됨
* 데이터를 수정할 때 조건절에서 검색되는 행 수만큼 수정된다.



* 만약 empno가 100번인 직원이 2명이면 2명의 ename이 모두 ‘조조’로 수정됨

DELETE

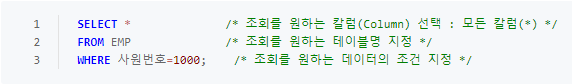
* 원하는 조건을 검색해서 해당되는 행 삭제
* 조건문 입력하지 않으면 모든 데이터 삭제
* DELETE 문으로 데이터를 삭제하면 용량이 감소할 것으로 생각 되는데, DELETE 문은 삭제 여부만 표시하고 용량은 초기화되지 않는다.



* WHERE 절에 조건을 입력하지 않으면 <EMP>테이블의 모든 데이터가 삭제 된다.

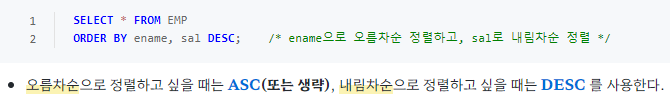
SELECT

* 테이블에 입력된 데이터를 조회하기 위해서 SELECT 문을 사용한다.
* SELECT문은 특정 칼럼이나 특정 행 만을 조회할 수 있다.



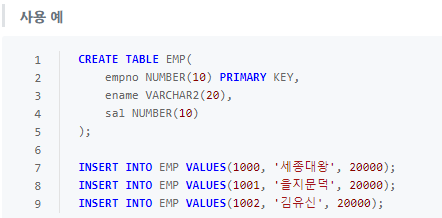
* 단, WHERE 절에 있는 조건문에 있는 행만 조회한다.

ORDER BY를 사용한 정렬



INDEX를 사용한 정렬 회피

* 정렬은 Oracle 데이터베이스에 부하를 주므로, 인덱스를 사용해서 ORDERBY를 회피 가능



* 위와같이 데이터를 입력하고 SELECT문을 실행하면 empno로 오름차순 정렬
* Empno가 기본키이기 때문에 자동으로 오름차순 인덱스가 생성

