Database

1. 데이터베이스란?

데이터베이스는 여러 사람들이 공동적으로 관리하는 데이터를 모아둔 집합체이다.

1-1. 파일시스템의 문제점

예전에는 클라이언트가 서버에게 요청을 하면 서버는 해당 파일을 open()하거나 write() 과정을 일일이 거쳐야했다. 하지만 여러 사람들이 동시에 접근하는 파일의 경우에는 중복되어 생성되는 경우도 많았으며 한곳에서만 데이터가 수정될 경우 불일치성의 문제가 생길 수 있다.

1-2. DBMS

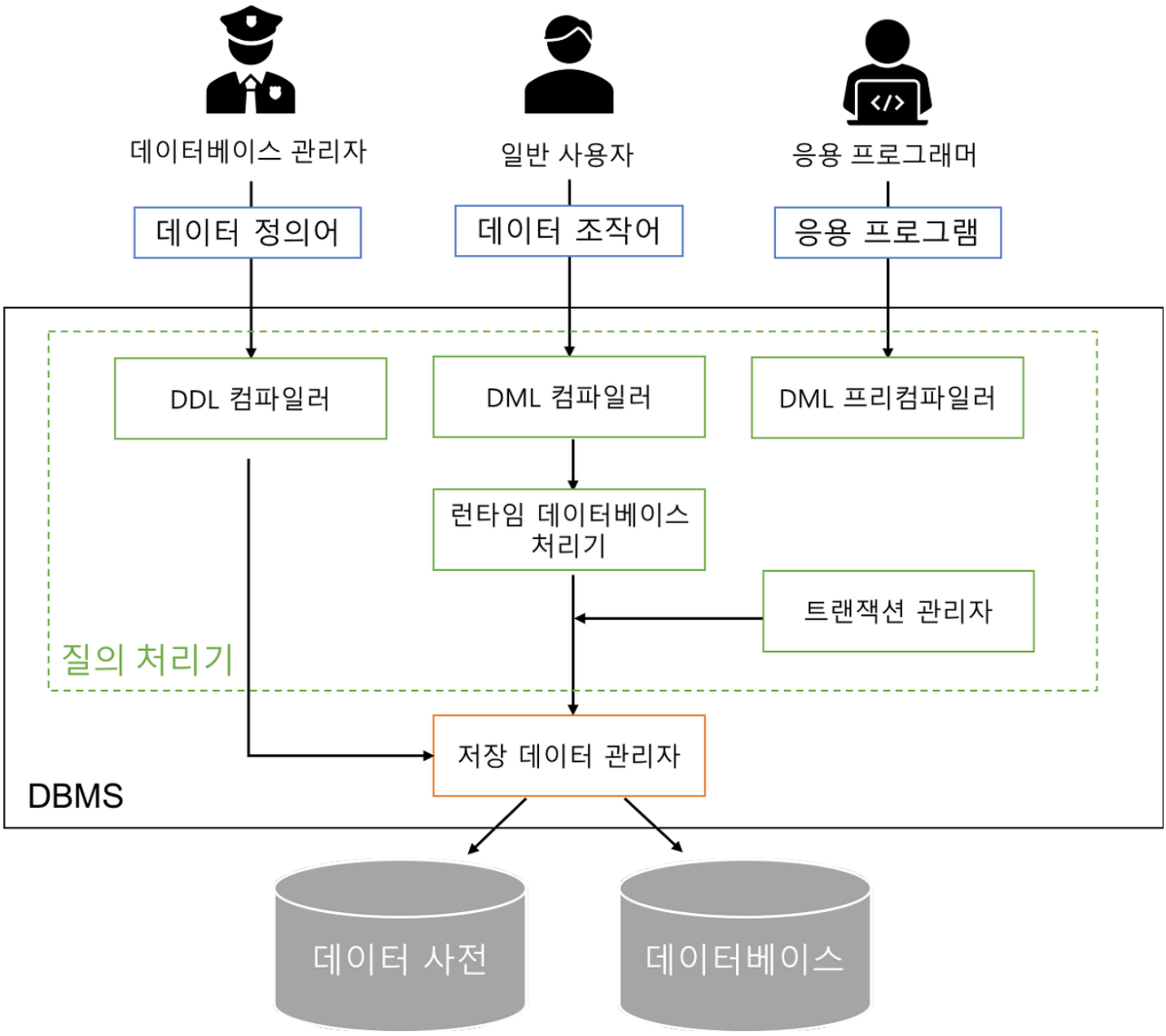
파일시스템의 문제를 해결하기 위해서 DBMS로 데이터를 관리한다. 데이터베이스에 저장될 데이터의 중복성과 관련 데이터 검색 제어등을 담당하는 시스템이다. DBMS는 Data Language를 사용해야하는데 SQL이라고한다.

DBMS의 종류로는 Oracle, MySQL 등이 있다.

1-3. Data Language

- DDL (데이터 정의어) : Create, Alter, Drop
- DML (데이터 조작어) : Select, Insert, Delete, Update
- DCL (데이터 제어어) : Commit, Rollback, Grant, Revoke

1-4. DBMS 구조



https://img1.daumcdn.net/thumb/R1280x0/?scode=mtistory2&fname=https%3A%2F%2Fblog.kakaocdn.net%2Fdn%2F9QHJh%2Fbtqv7O348Ud%2FChBPHpTRWHo0IYA8OlYuq0%2Fimg.png

2. Key

2-1. Primary Key (기본키)

기본키란 데이터 테이블에서 고유한 값을 가진 속성값이다. NULL값을 허용하지 않으며 변경될 가능성이 높은것은 기본키로 설정하지 않는다.

테이블당 기본키는 1개만 구성할 수 있다.

- 유일성 : 하나의 키 값으로 하나의 튜플을 유일하게 식별 가능
- 최소성 : 키를 구성하는 속성 하나가 제거되면 유일하게 식별 불가하도록 최소한의 유일속성구성을 이루어야함.

(기본키의 속성이 2개이상이면 복합키라고 하지만 복합키는 기본키로 가능한 사용하지 않는것이 좋다)

2-2. Candidate Key (후보키) / Alternate Key (대체키)

테이블에는 튜플 1개만을 유일하게 특정지을 수 있는 속성값이 여러개 존재할 수도 있다. 예를들어, User 정보 테이블이라고 생각을 해보자. User 테이블에는 User의 이름, 주민번호, 아이디 속성이 있다.

아이디와 주민번호는 User를 구분할 수 있는 고유값이 될 수 있다. 하지만 우리는 아이디를 기본키로 정의한다.

그 이유는 유저 아이디는 요청 쿼리에 많이 사용되지만 주민번호는 아이디 분실시 아이디를 찾는 용도로 사용하기때문에 요청 쿼리 빈도가 작다. 유저 아이디와 주민번호는 고유 속성값인 후보키이지만 기본키는 아이디이므로 주민번호는 대체키가 된다.

- 후보키 : 아이디, 주민번호
- 기본키 : 아이디
- 대체키 : 주민번호 (후보키-기본키)

2-3. Foreign Key (외래키)

외래키는 기본키와 참조관계에 있는 키이다.

예를들어, 학생 수강 신청 관리 테이블이 있다고 해보자.(Student)

Student 테이블에는 속성값이 학번, 이름, 수강과목, 점수가 있고 기본키는 학번이다.

Subject 테이블에는 속성값이 수강과목, 교수님, 강의실이 있고 기본키는 수강과목이다.

이때 클라이언트가 학생들이 수업을 들을 강의실이 궁금하다하면 Student의 수강과목을 외래키로 하여 Subject 테이블을 검색하여 강의실을 찾는다.

외래키의 장점은 어느 한 과목의 강의실이 바뀔 경우 그 수업을 듣는 모든 학생들의 강의실 정보를 수정하지 않고 Subject 테이블에서 해당 과목 강의실 정보만 바꾸어도 되는 장점이 있다. (데이터 무결성)

- 외래키를 사용하지 않는 경우가 더 많다. 이는 왜그럴까 ?

  이는 성능때문에 사용을 꺼린다고한다.

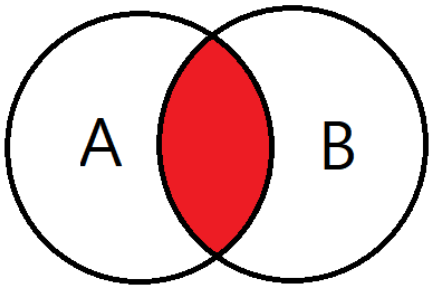
  1. 부모테이블(Subject)에서 자료를 삭제할때 자식테이블(Student)에 영향을 주는지 모두 확인해야한다. 만약 자식테이블에서 부모테이블 정보가 있다면 이를 다 삭제하고 삭제할 수 있다. (비용발생)
  2. 자식테이블에 새로운 정보를 추가할때 부모테이블에 정보가 있는지 확인해야한다.
  3. 자식테이블에서 외래키를 update할때 부모테이블을 확인해야한다.

  > https://okky.kr/article/586565

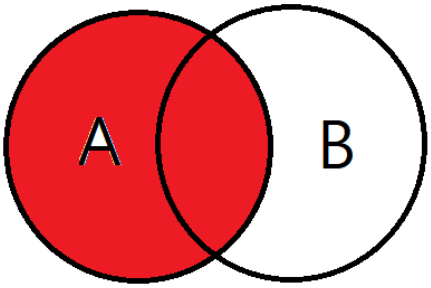
3. JOIN

> https://coding-factory.tistory.com/87

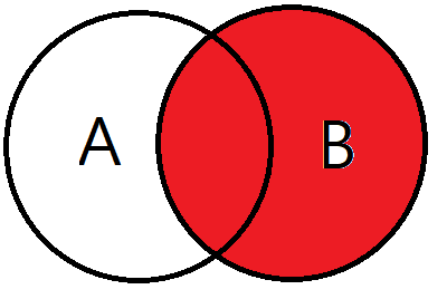
3-1. INNER JOIN



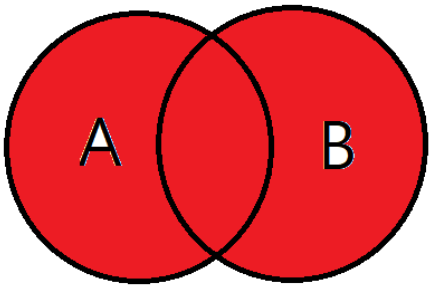
3-2. LEFT OUTER JOIN



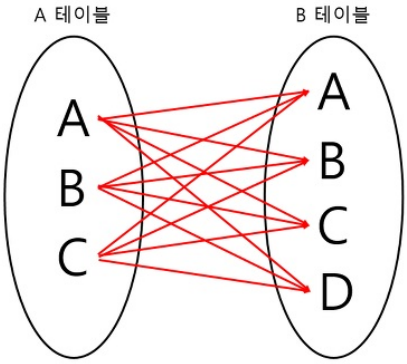
3-3. RIGHT OUTER JOIN



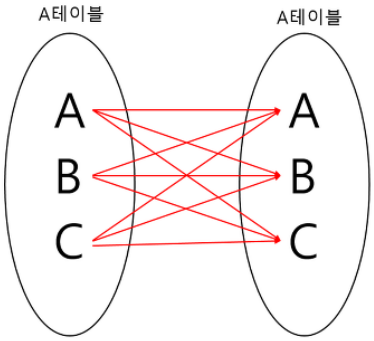
3-4. FULL OUTER JOIN



3-5. CROSS JOIN



3-6. SELF JOIN



4. Transaction

트랜잭션이란 작업의 안정성을 보장해주는 것이다. 트랜잭션의 특성은 ACID이다.

- Atomicity : 원자성
- Consistency : 일관성
- Isolation : 고립성
- Durability : 지속성

4-1. ACID

- 원자성 : 작업 중간에 문제 발생 없을 경우에만 수행한다. (모두 반영되거나, 모두 반영되지 않거나)
- 일관성 : 데이터가 업데이트 후에도 DB제약조건을 만족시키고 있어야한다. (전상황과 동일해야함)
- 고립성 : 각각의 트랜잭션은 서로 간섭없이 독립적으로 수행된다.
- 지속성 : 트랜잭션이 정상으로 종료되어도 영구적으로 데이터베이스에 작업의 결과가 저장되어있어야한다.

4-2. 트랜잭션 예시

가장 예시로 많이 들어지는 것은 돈거래이다. 예를들어 A가 B에게 10000을 송금한다고 해보자.

- A : 10000 출금
- B : 10000 입금

이 둘이 동시에 이루어진다면 ? 문제가 발생!

A가 먼저 수행되고 B가 수행되어야한다. 이러한 관련 쿼리를 묶어서 수행하는 것을 트랜잭션이라고 한다.

1. 출금 쿼리 처리기 (10000 출금)
2. 데에터 캐시가 필요한 데이터 파일 search
3. 로그 캐시에 undo 기록 : 변경 전의 값을 기록
4. 로그 캐시에 redo 기록 : 변경 후의 값을 기록
5. 데이터 캐시에 결과 반영

로그 캐시를 사용하는 이유는 만약 예상치못한 오류가 발생했을 경우를 대비해서이다.

오류 발생시 Redo를 먼저 수행하고 Undo를 실행한다.

ROLLBACK이 발생했을 경우에는 Undo를 실행한다.

4-3. 트랜잭션 격리 수준

동시에 데이터베이스에 접근하는 경우 제어방버을 말한다.

1. Read-Uncommited
2. Read-Commited
3. Repeatable-Read
4. Serializable

1->4로 갈수록 격리가 높아지고 4->1로 갈수록 동시성을 허용한다.

1. 커밋전의 트랜잭션의 데이터 변경 내용을 다른 커밋이 읽게 허용해준다.

   -> Dirty Read 발생!!!!!

   -> 만약 A가 바꾼 것을 B가 읽었는데 A가 롤백하면 문제가 생긴다.

2. 커밋이 완료된 것만 다른 트랜잭션이 읽게해준다.

   -> A 트랜잭션이 수행중이면 변경 전 것만 B가 읽을 수 있다.

   -> A 트랜잭션이 커밋하면 변경후인 데이터를 B가 읽을 수 있다.

   -> Non-Repeatable Read 발생!!!!!

   -> B트랜잭션이 select를 두번하면 A 커밋전과 커밋후의 내용이 달라져버릴 수 있다.

3. 트랜잭션 범위 내에서 조회한 내용이 항상 동일하도록 보장시킨다.

   -> A 트랜잭션이 커밋전에 B 트랜잭션이 읽었으면 A 트랜잭션이 커밋후에도 B 트랜잭션은 A의 커밋 전값을 읽는다.

   -> Phandom Read 발생!!!! (Non-Repeatable Read 종류)

   -> 새로 생성된 행이나 없어진 행은 반영이 안된다. (커밋전이나 커밋후가 달라져 버림...)

4. 완전 독립성 보장, 다른 트랜잭션이 접근 절대 불가.