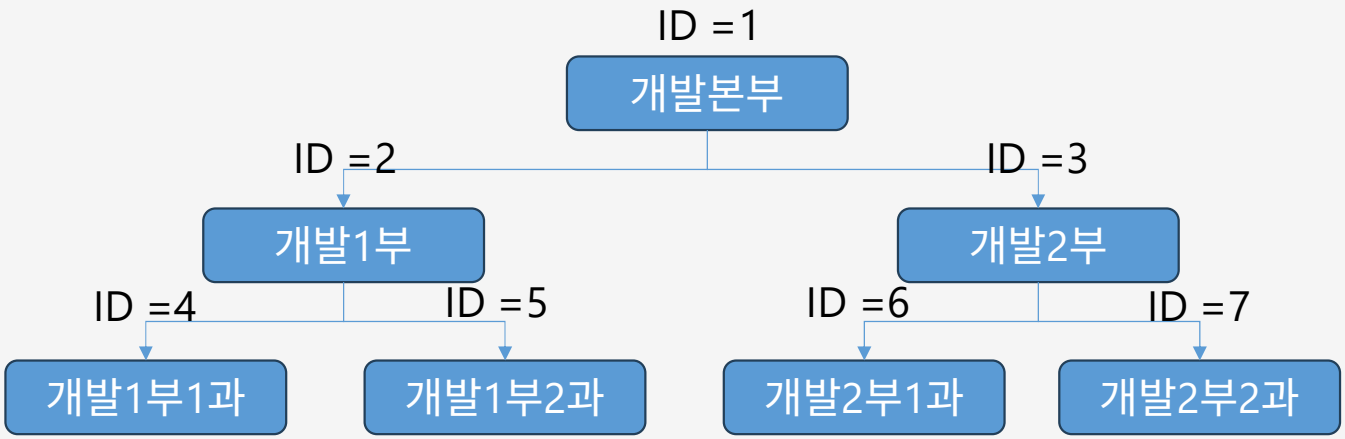
트리구조 테이블에 관한 고민

트리구조란?

데이터의 계층구조를 나무와 같은 형태로 표현한 것.트리란 데이터형태의 한 종류로 계층적인 부모-자식관계를 표현할 때 자주 사용됨.

그러나, 계층적인 부모-자식관계를 표현할 때 자주 사용된다는 인식때문에 생각없이 부모-자식관계 = 트리구조로 결정! 같은 일이 벌어지는 경우가 많으나 잘못된 생각일지도 모른다는 의문이 들기 시작했다.



하지만?

설명:트리형태의 테이블을 작성했을 때의 문제점

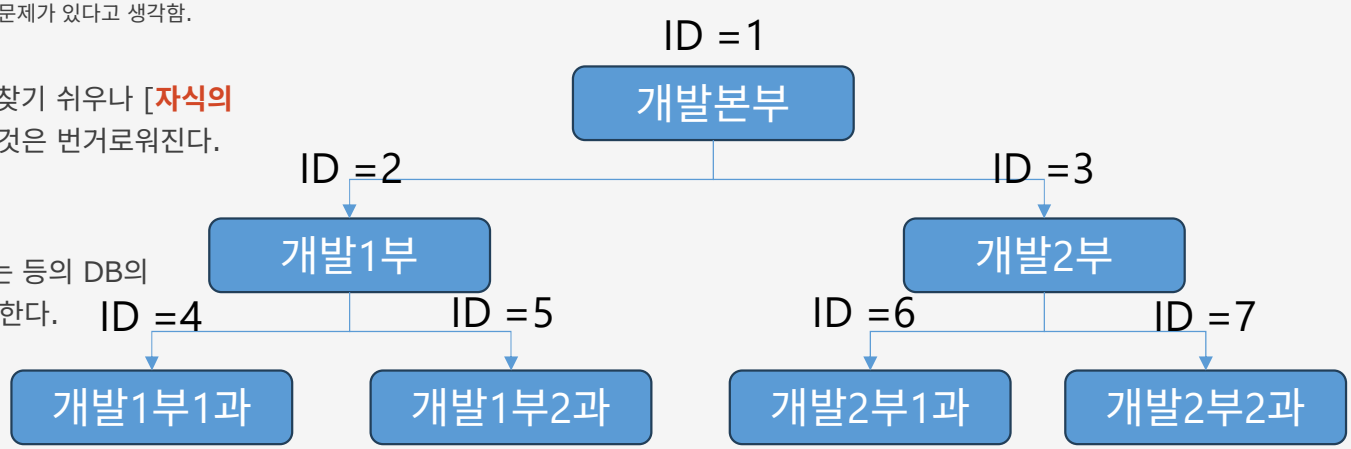
1. 트리는 특정 레코드를 빠르게 [**검색**]하기 위한 형태라는 것을 이해하고서,
[**부서**]같은 데이터가 가지는 특성과 거기서 파생되는 요건을 생각해 볼 필요가 있다.

2. 자주 있는 예시로서, [**최상위부서**] 를 기점으로 [**하위부서 전체**]를 표시하고 싶다는 요건이 있는데,
이런 요건이 발생할 경우 트리 구조는 SQL을 짜기가 어렵다.

해설: 종종 재귀적인 데이터 구조를 가진 테이블을 위한 재귀SQL이 존재하기는 하나 잘못된 데이터 구조를 DB의 기능이나 비지니스 로직으로 커버한다는 것은 애초에 문제가 있다고 생각함.

3. [**자식레코드**] ,[**부모레코드**]에 해당하는 데이터는 찾기 쉬우나 [**자식의 자식**] 、[**부모의 부모**]에 해당하는 레코드를 찾는 것은 번거로워진다.

4. 검색속도가 느려지면서 [**데이터 전체를 캐싱**]하자는 등의 DB의 존재의의를 무의미하게 만드는 대응이 발생하기도 한다.



제안1:트리구조를 포기한다

해설:트리구조를 포기하고 부모의 수만큼 컬럼을 추가한다.

1. SQL을 짜기 편해진다.

2. 한번의 검색으로 모든 부모를 검색할 수 있기때문에[**검색속도**] 이 향상된다.

문제점

1. 부모의 수를 늘리고 싶어졌을 경우, DB만이 아니라 어플리케이션까지 수정해야하는 필요성이 생기기 때문에 [**확장성**]의 문제가 생긴다.

2. [**조직변경**] 과 같은데이터갱신이 발생했을 경우 처리비용이 증가한다.

3. [**데이터 무결성**]을 DB기능없이 담보해야하는 문제가 생긴다.

ID	NAME	PARENT
1	개발본부	NULL
2	개발1부	1
3	개발2부	1
4	개발1부1과	2
5	개발1부2과	2

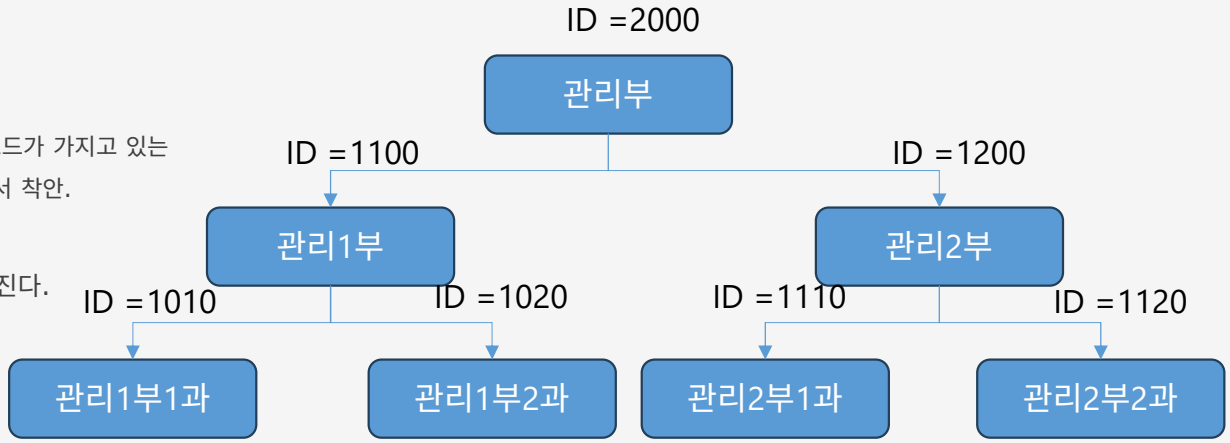
ID	NAME	P1	P2	P3
1	개발본부	NULL	NULL	NULL
2	개발1부	1	NULL	NULL
3	개발2부	1	NULL	NULL
4	개발1부1과	2	1	NULL
5	개발1부2과	2	1	NULL

제안2:테이블을 좀 더 트리구조에 맞게 변형한다

설명:ID만으로 일련의 데이터군을 지정할 수 있도록 ID를 부여한다.

1. 트리는 단순히 나무와 같은 형태의 데이터를 가리키는 것이 아니라 각 노드가 가지고 있는 값을 사용하여 [**트리구조로 정렬**]하고、[**검색을 용이**]하게 한다는 것에서 착안.

2. [**일련의 데이터군**] 을[**일정한 범위**]의 값을 사용해 검색이 간편해진다.
   개발본부=ID <= 1000의부서
   관리부=ID >1000 and ID <=2000의 부서
   관리1부=ID <= 100의 부서



문제점

1. ID를 생성하는 로직을 [**작성할 필요**]가 있는 것과 ID를 생성하는 로직에 문제가 있을 경우 [**버그로 직결**]되는 문제가 있다.

2. [**테이블의 구조 및 사용법**]을 테이블 정의만 보고서는 [**파악 할 수 없다**]는 문제가 있다.