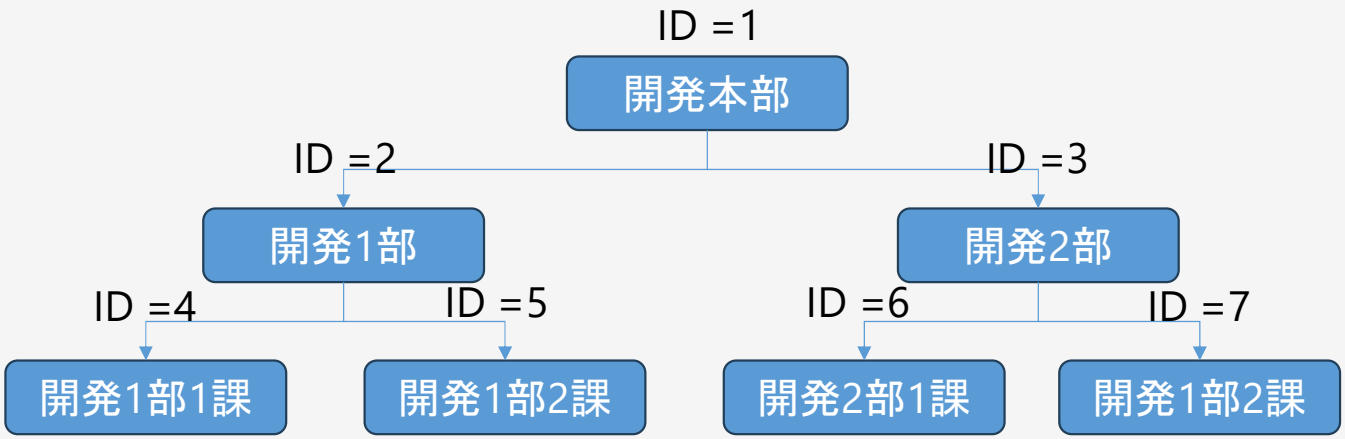
Tree構造のテーブルに対する悩み

ツリー構造とは？

データの階層構造を木のような形で表現したもの。
ツリー構造とは、データ構造の一種で、階層的な親子関係を持つ要素を表現する際に用いられる。

しかし、階層的な親子関係を持つ要素を表現する際に用いられるという認識のせいで、考え無に親子を持ってる＝ツリー構造決定！になることが多いが、間違った判断かも知れないと最近感じ始めた。



しかし？

説明:ツリー構造をテーブル化した時の問題

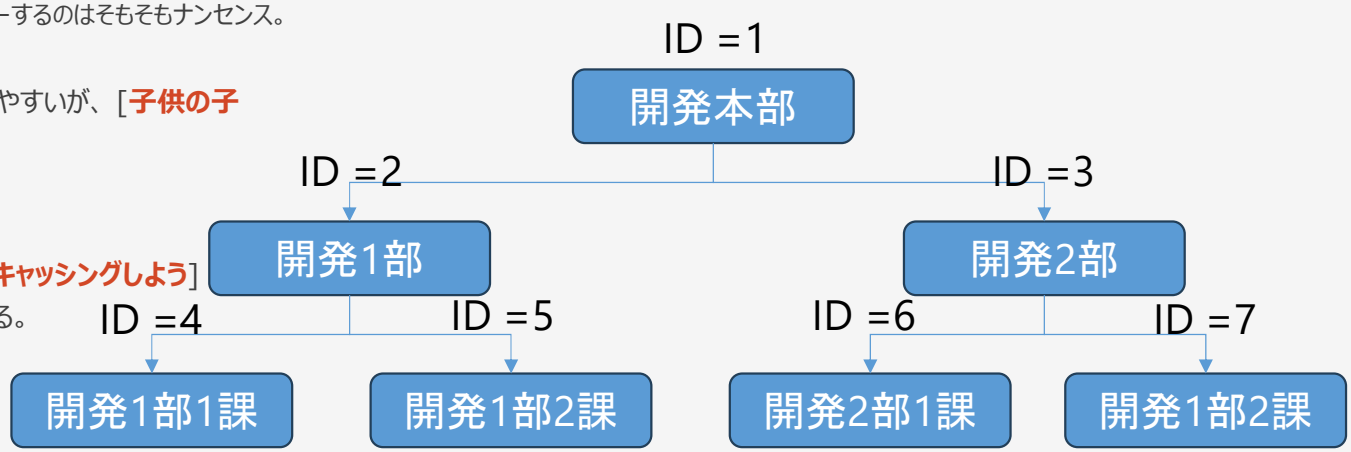
1. ツリー構造は特定のレコードを早く [**検索**]するため作法ということを理解した上で [**部署**]のようなデータが持つ特性とそこから派生する仕様を考える必要がある。

2. 良くある事例は、 [**最上位部署**] を起点に[**下位の部署を全部**] 表示したいという要件。こういう要件が発生した場合、SQLが組みづらい。

   補足: DBによっては再帰的な構造のデータのための再帰SQLというものが存在はするが、間違ったデータ構造をDBの機能や、ビジネスロジックでカバーするのはそもそもナンセンス。

3. [**自分の子供**] 、[**自分の親**]に該当するデータは探しやすいが、 [**子供の子供**] 、 [**親の親**] を探すというのはコストが大きい。

4. 検索のコストが大きくなって、DBを漁らないで、 [**全件キャッシングしよう**] などのDBの存在価値がない対応が生まれる恐れがある。



案１：ツリー構造をやめる

説明:ツリー構造を止めて親のの数分カラムを持たせる。

1. SQLが組みやすくなる。

2. 一発の検索ですべての親を取得出来るため、[**検索のパフォーマンス**] 側面が向上する。

問題点

1. 親の数を増やしたいなどの修正が発生した場合、DBだけではなく、アプリケーションまで修正が及ぶため、[**拡張性**]の問題を抱えている。

2. [**組織変更**] などのデータの更新が発生した時に、更新のコストが大きくなる。

3. [**整合性**]をDBの機能なしでビジネスロジックで実現しないといけない。

ID	NAME	PARENT
1	開発本部	NULL
2	開発1部	1
3	開発2部	1
4	開発1部1課	2
5	開発1部2課	2

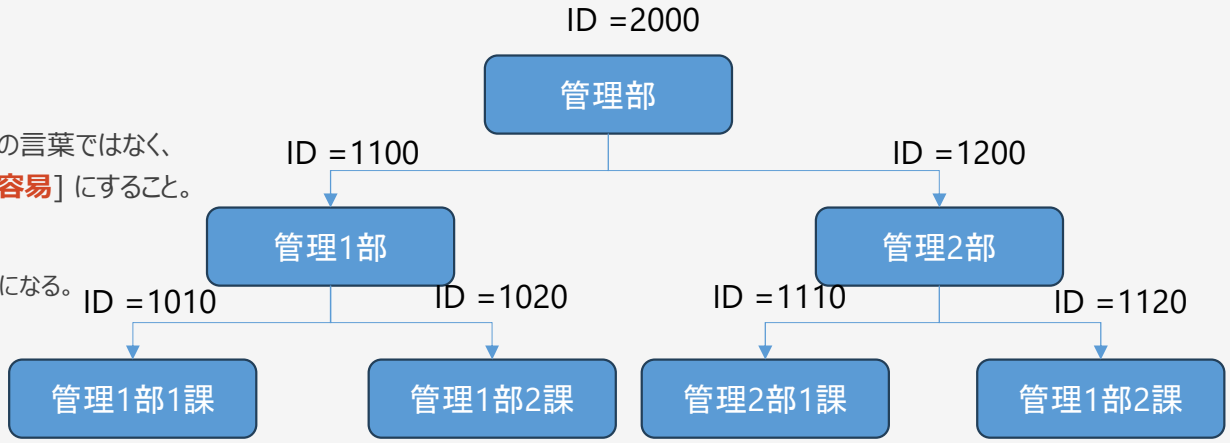
ID	NAME	P1	P2	P3
1	開発本部	NULL	NULL	NULL
2	開発1部	1	NULL	NULL
3	開発2部	1	NULL	NULL
4	開発1部1課	2	1	NULL
5	開発1部2課	2	1	NULL

案２：テーブルをもう少しツリー構造に寄せる

説明:IDだけで一連の群れのデータを見分ける様にIDを付与する。

1. ツリーとはそもそもただ単にピラミッドみたいな形のデータを指すだけの言葉ではなく、各ノードが持っている値から、[**ツリー構造へソート**]し、[**検索を容易**] にすること。

2. [**一連のデータ群**] を[**一定の範囲**] の値に留めることで、検索が容易になる。
   開発本部=ID <= 1000の部署
   管理部=ID >1000 and ID <=2000の部署
   開発1部=ID <= 100の部署



問題点

1. IDを生成するロジックを自前で[**作成する必要**]があることと、IDを生成するロジックに問題が生じるとシステム全体の [**バグに直結**]する。

2. [**テーブルの仕様**] をテーブルを見て[**把握出来ない**]という問題がある。