06 数据库设计（一致性保证）

数据库范式

- 数据库的设计范式
    - 1NF：属性的原子性
    - 2NF：属性的主键完全依赖
    - 3NF：不存在传递函数依赖
    - BCNF、4NF、5NF（完美范式）

关系模式$R \in 1NF$,如果对于R对于R的每个非平凡多值依赖$X \to\to Y$(Y不属于X),X都含有候选码，则$R \in 4NF$。

- 直到关系理论出现，数据库设计是“科学（science）”而非“工艺（craft）”

数据库反范式

- 反范式，为什么?
  - 规范化的结果是一个在结构上一致，且拥有最少冗余的逻辑数据库设计
  - 但未必是性能最优的设计
- 反范式本质是——考虑引入可控制的冗余（Controlled Redundancy）
  - 实现更加复杂（你需要手动保持数据的一致性）
  - 降低灵活性（通用性和灵活性，通用性一致且简单，灵活性固定且高效）
  - 往往加快了读取，降低了更新

数据库反范式

- **反范式，最需要的动作是什么?**

增加冗余，最主要的动作就是复制，将属性和对属性们的统计处理复制到其他的地方。

- **反范式，最核心的目标是什么?**

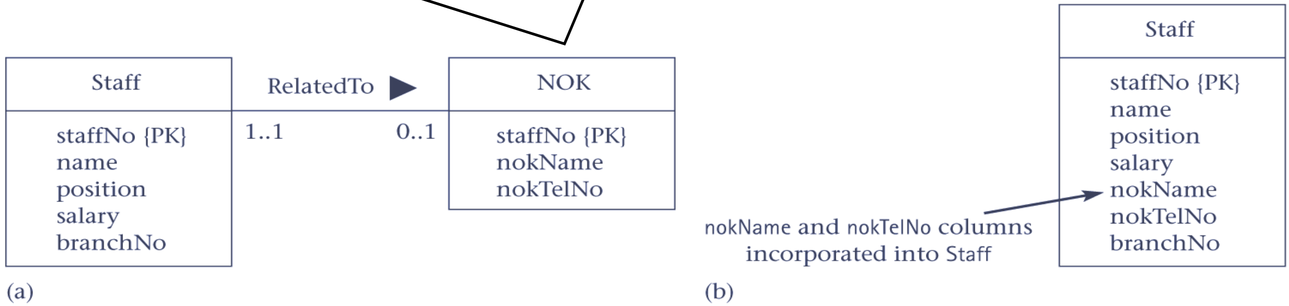
复制的根本目标是什么?降低连接次数，记住，几乎所有的反范式的基本逻辑，就是通过降低连接提高效率。

数据库反范式模式（Pattern）

- Pattern1：合并 1:1 关系
- Pattern2：复制 1:* 关系的非 Key 、FK及值
- Pattern3：复制 *:* 关系的属性
- Pattern4： 引入重复组
- Pattern5： 创建提取临时表

Pattern1：合并 1:1 关系

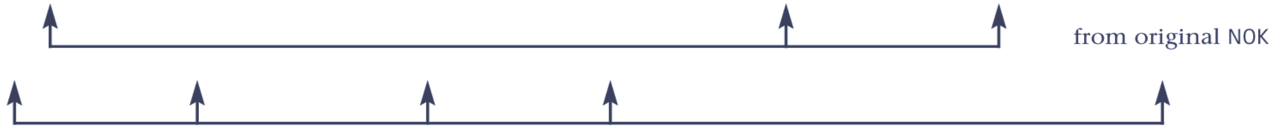
部分参与会产生大量 NULL，大量空间浪费。



最复杂的问题是两边都是部分参与，该怎么办?

Staff

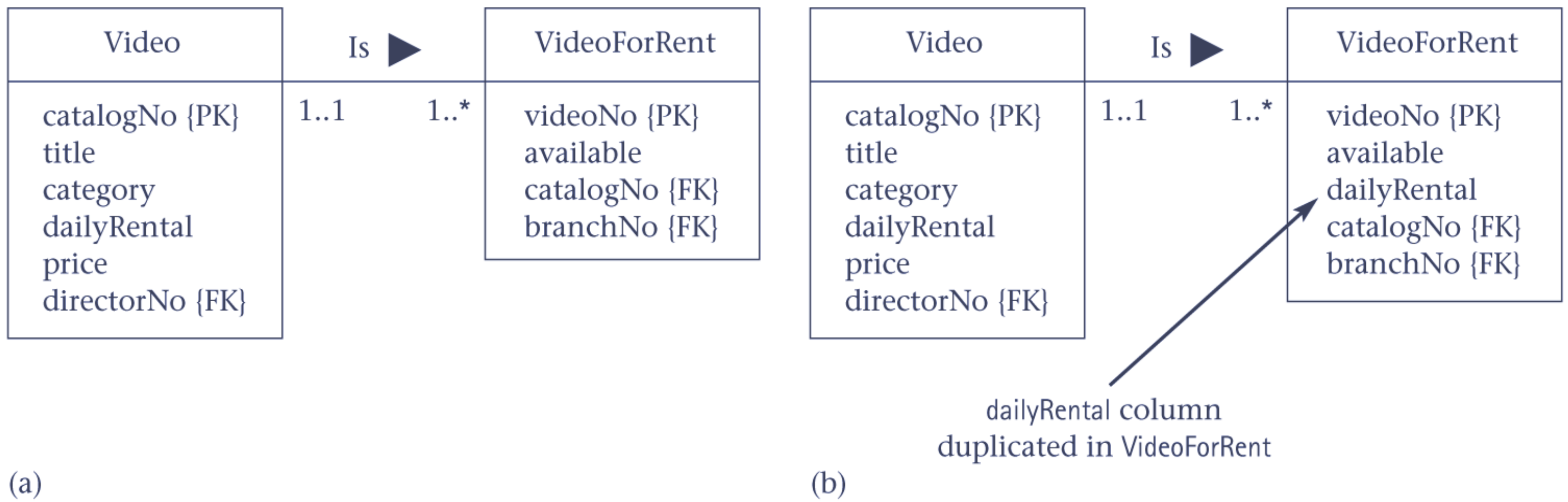
staffNo	name	position	salary	nokName	nokTelNo	branchNo
S1500	Tom Daniels	Manager	46000	Jane Daniels	207-878-2751	B001
S0003	Sally Adams	Assistant	30000	John Adams	518-474-5355	B001
S0010	Mary Martinez	Manager	50000			B002
S3250	Robert Chin	Supervisor	32000	Michelle Chin	206-655-9867	B002
S2250	Sally Stern	Manager	48000			B004
S0415	Art Peters	Manager	41000	Amy Peters	718-507-7923	B003



from original Staff

(c)

Pattern2：复制 1:* 关系的非 Key 和 FK



Video
catalogNo {PK}
title
category
dailyRental
price
directorNo {FK}

Is ▶ 1..1 1..*

VideoForRent
videoNo {PK}
available
catalogNo {FK}
branchNo {FK}

(a)

Video
catalogNo {PK}
title
category
dailyRental
price
directorNo {FK}

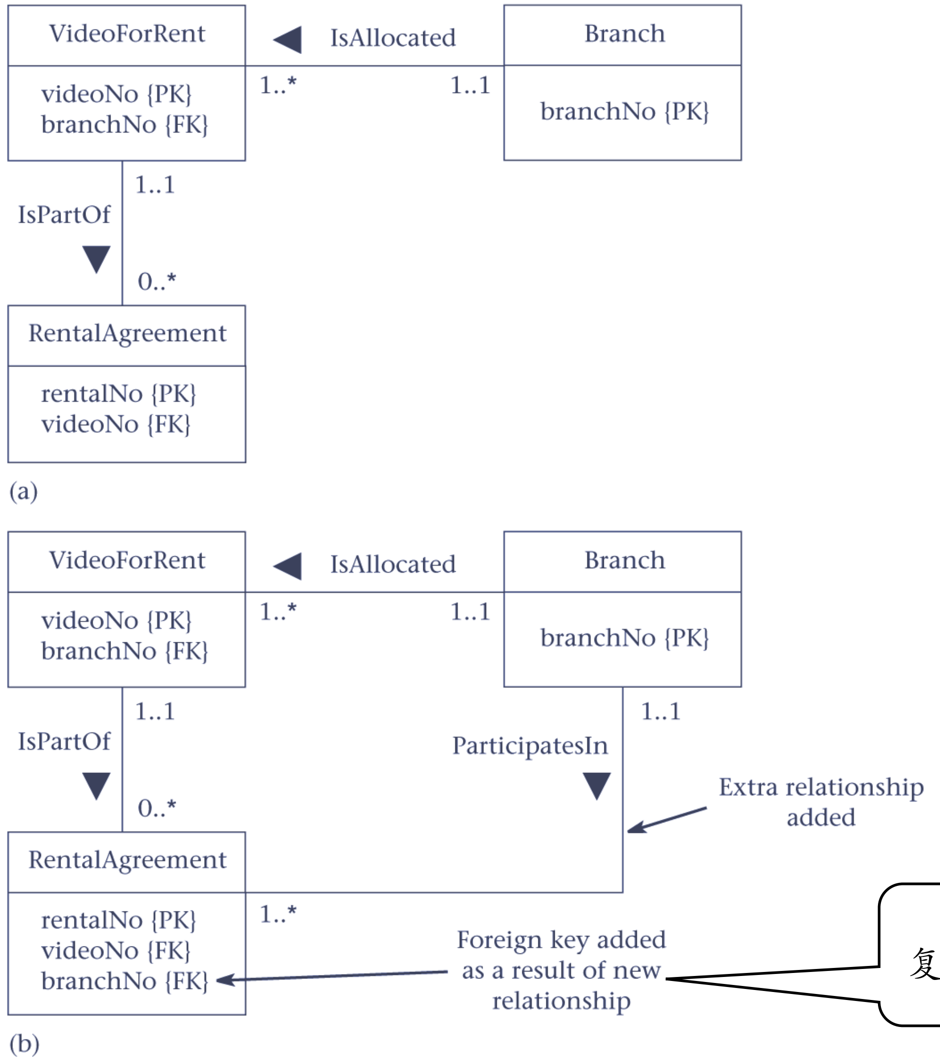
Is ▶ 1..1 1..*

VideoForRent
videoNo {PK}
available
dailyRental
catalogNo {FK}
branchNo {FK}

dailyRental column
duplicated in VideoForRent

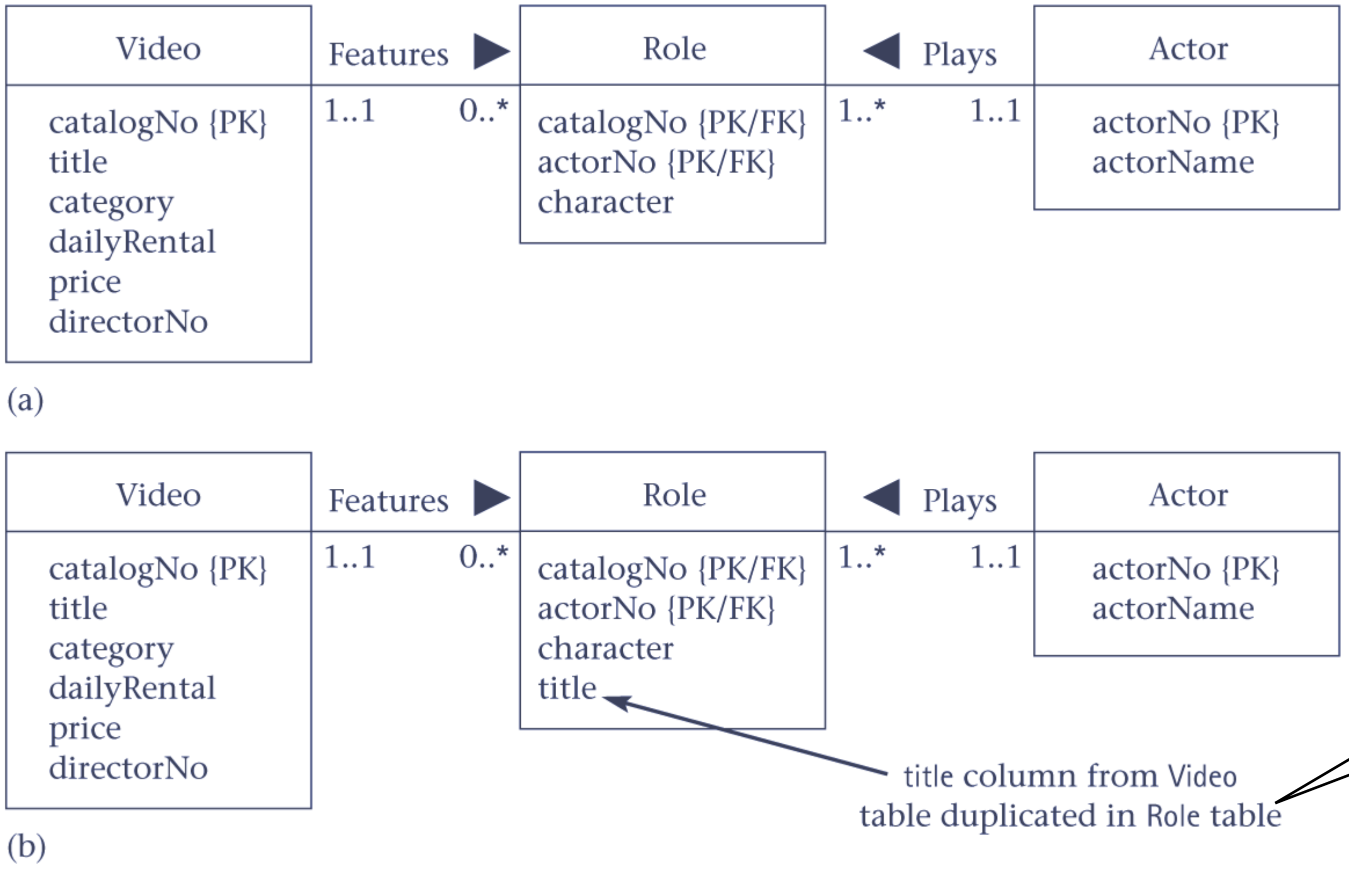
(b)

Pattern2：复制 1:* 关系的非 Key、FK及值



复制的最常见的，是直接复制值，而切断连接本身

Pattern3：复制 *:* 关系的属性



在多对多关系表中，增加属性、属性的统计信息的新增字段，是业内非常常见的逻辑。

数据库反范式模式（Pattern）

- Pattern4： 引入重复组
  - 地址、电话
  - 静态、数量较小
- Pattern5： 创建提取临时表
  - 这是 DB 开发者中最大的毒药
  - 静态，时间切片，不是实时数据
  - 实时计算，就变成物化视图（实时更新的巨大压力，可能得不偿失）
  - 一个特别好用的、特别好吃的、特别理想的技术、物品、人，都是危险的

数据库的设计

- 同样一个目标，满足 3NF 可以有多种数据模式设计
- 不同的数据库设计
  - 完成的功能细节不一样
  - 相同目的的 SQL，效率会有巨大差异
  - 比如——12306 的表结构设计

处理层次结构（Hierarchical Data）

- 树状结构（Tree Structures）
  - 历史…
    - 层次数据库
    - 网状数据库
    - 关系型数据库
  - 直到关系理论出现，数据库设计是“科学（science）”而非“工艺（craft）”
    - 层次性数据广泛存在（XML，LDAP，BOM…）
  - 层次结构复杂度在于
    - 访问树的方式

树状结构VS.主从结构

- 父子结构（parent/child link）--tree structure
- 主从结构（master/detail relationship）
- 差异
  - 树状结构保存只需要一张表
  - 深度
  - 所有权
  - 多重父节点

Fabian Pascal：Practical Issues in Database Management（Addion Wesley）

层次结构的实际案例

- Risk exposure
- 档案位置
- 原料使用
- ……
- 不同的案例具有不同的基本特征
- 通常，树中的节点数量偏小。实际上，这也是树的优点，便于高效检索

层次结构的实际案例

```
select building.name building,
        floor.name floor,
        room.name room,
        alley.name alley,
        cabinet.name cabinet,
        shelf.name shelf,
        box.name box,
        folder.name folder
   from inventory,
       location folder,
       location box,
       location shelf,
       location cabinet,
       location alley,
       location room,
       location floor,
       location building
   where inventory.id = 'AZE087564609'
    and inventory.folder = folder.id
    and folder.located_in = box.id
    and box.located_in = shelf.id
    and shelf.located_in = cabinet.id
    and cabinet.located_in = alley.id
    and alley.located_in = room.id
    and room.located_in = floor.id
    and floor.located_in = building.id
```

用SQL数据库描述树结构

- 只要对象的类型相同，而对象的层树可变，其关系就应该被建模为树结构
- 在数据库设计中，树通常三种模型
  - Adjacency model-邻接模型
  - Materialized path model-物化路径模型
  - Nested set model-嵌套集合模型
    - Joe Celko发明
    - Vadim Tropashko 提出过nested interval model

数据来源http://www.kessler-web.co.uk

树的实际实现：邻接模型

数据库设计的归一化原则：一事、一地、一次

（one simple fact 、 in one place，on one time）

ADJACENCY_MODEL

Name	Null?	Type
ID	NOT NULL	NUMBER
PARENT_ID		NUMBER
DESCRIPTION	NOT NULL	VARCHAR2(120)
COMMANDER		VARCHAR2(120)

表的每一行描述一个部队，parent_id指向树中的上级部队

来吧，说说这个模型有什么问题?

```
 ID    PARENT_ID    DESCRIPTION                     COMMANDER
---    ---------    ----------------------------    ----------------------------
435      0          French Armée du Nord of 1815    Emperor Napoleon Bonaparte
619     435         III Corps                       Général de Division Dominique
                                                    Vandamme
620     619         8th Infantry Division           Général de Division Baron
                                                    Etienne-Nicolas Lefol
621     620         1st Brigade                     Général de Brigade Billard
                                                    (d.15th)
622     621         15th Rgmt Léger                 Colonel Brice
623     621         23rd Rgmt de Ligne              Colonel Baron Vernier
624     620         2nd Brigade                     Général de Brigade Baron
                                                    Corsin
625     624         37th Rgmt de Ligne              Colonel Cornebise
626     620         Division Artillery
627     626         7/6th Foot Artillery            Captain Chauveau
```

树的实际实现：物化路径模型

它能解决归一化的问题吗?

MATERIALIZED_PATH_MODEL

Name	Null?	Type
MATERIALIZED_PATH	NOT NULL	VARCHR2(25)
DESCRIPTION	NOT NULL	VARCHAR2(120)
COMMANDER		VARCHAR2(120)

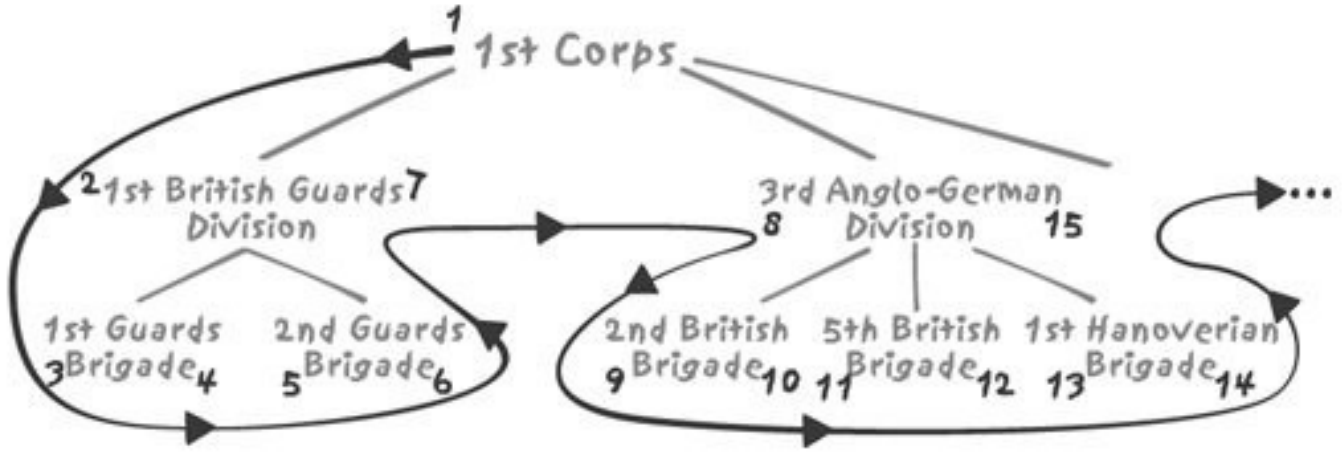
表中有两个索引，在materialized_path上的唯一性索引以及在commander上的索引，正确的设计应该增加id字段。

```
 MATERIALIZED_PATH   DESCRIPTION                    COMMANDER
-----------------    ---------------------------   -------------------------
F                    French Armée du Nord of 1815   Emperor Napoleon Bonaparte
F.3                  III Corps                      Général de Division
                                                    Dominique Vandamme
F.3.1                8th Infantry Division          Général de Division Baron
                                                    Etienne-Nicolas Lefol
F.3.1.1              1st Brigade                    Général de Brigade Billard
                                                    (d.15th)
F.3.1.1.1            15th Rgmt Léger                Colonel Brice
F.3.1.1.2            23rd Rgmt de Ligne             Colonel Baron Vernier
F.3.1.2              2nd Brigade                    Général de Brigade Baron
                                                    Corsin
F.3.1.2.1            37th Rgmt de Ligne             Colonel Cornebise
F.3.1.3              Division Artillery
F.3.1.3.1            7/6th Foot Artillery           Captain Chauveau
```

树的实际实现：嵌套集合模型

NESTED_SETS_MODEL

Name	Null?	Type
DESCRIPTION		VARCHAR2(120)
COMMANDER		VARCHAR2(120)
LEFT_NUM	NOT NULL	NUMBER
RIGHT_NUM	NOT NULL	NUMBER



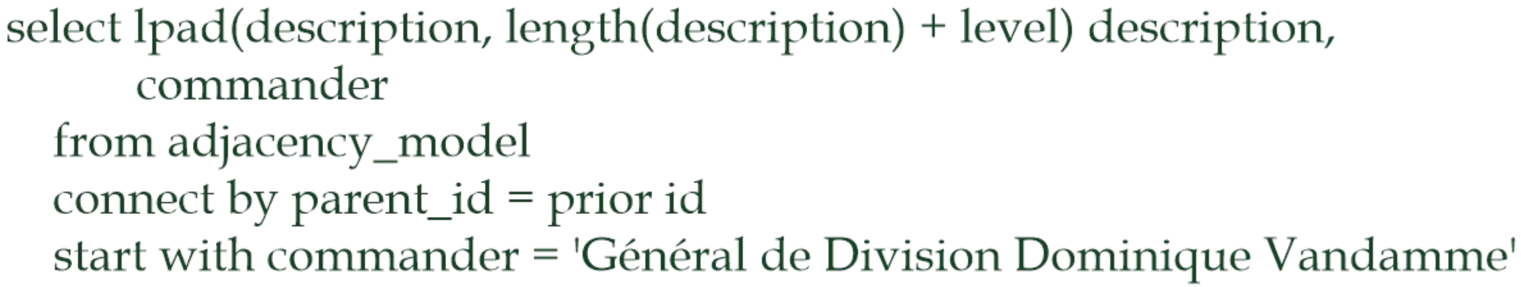
```
 DESCRIPTION                      COMMANDER                      LEFT_NUM    RIGHT_NUM
-----------------------------    ---------------------------    --------    ----------
Armies of 1815                                                   1           1622
French Armée du Nord of 1815     Emperor Napoleon Bonaparte     870         1621
III Corps                        Général de Division            1237        1316
                                 Dominique Vandamme
8th Infantry Division            Général de Division Baron      1238        1253
                                 Etienne-Nicolas Lefol
1st Brigade                      Général de Brigade Billard     1239        1244
                                 (d.15th)
15th Rgmt Léger                  Colonel Brice                  1240        1241
23rd Rgmt de Ligne               Colonel Baron Vernier          1242        1243
2nd Brigade                      Général de Brigade Baron       1245        1248
                                 Corsin
37th Rgmt de Ligne               Colonel Cornebise              1246        1247
Division Artillery                                              1249        1252
7/6th Foot Artillery             Captain Chauveau               1250        1251
```

用SQL访问树结构

- 为了检查效率和性能，分别用不同模型解决如下两个问题：
- 法国将军Dominique Vandamme指挥哪些部队，以缩排方式或简单列表的方式显示他们。注意，所有的commander字段都构建了索引（简称Vandamme查询）
- Scottish Highlanders的每个团各属于哪个部队（自底向上的查询）。在部队的名称（description字段）上没有索引，唯一的方法是在description字段中查找“Highland”字符串，在没有任何全文索引的情况下，这个问题简称highland问题
    - 注：层次结构Corp-division-brigade-regiment
    - Oracle

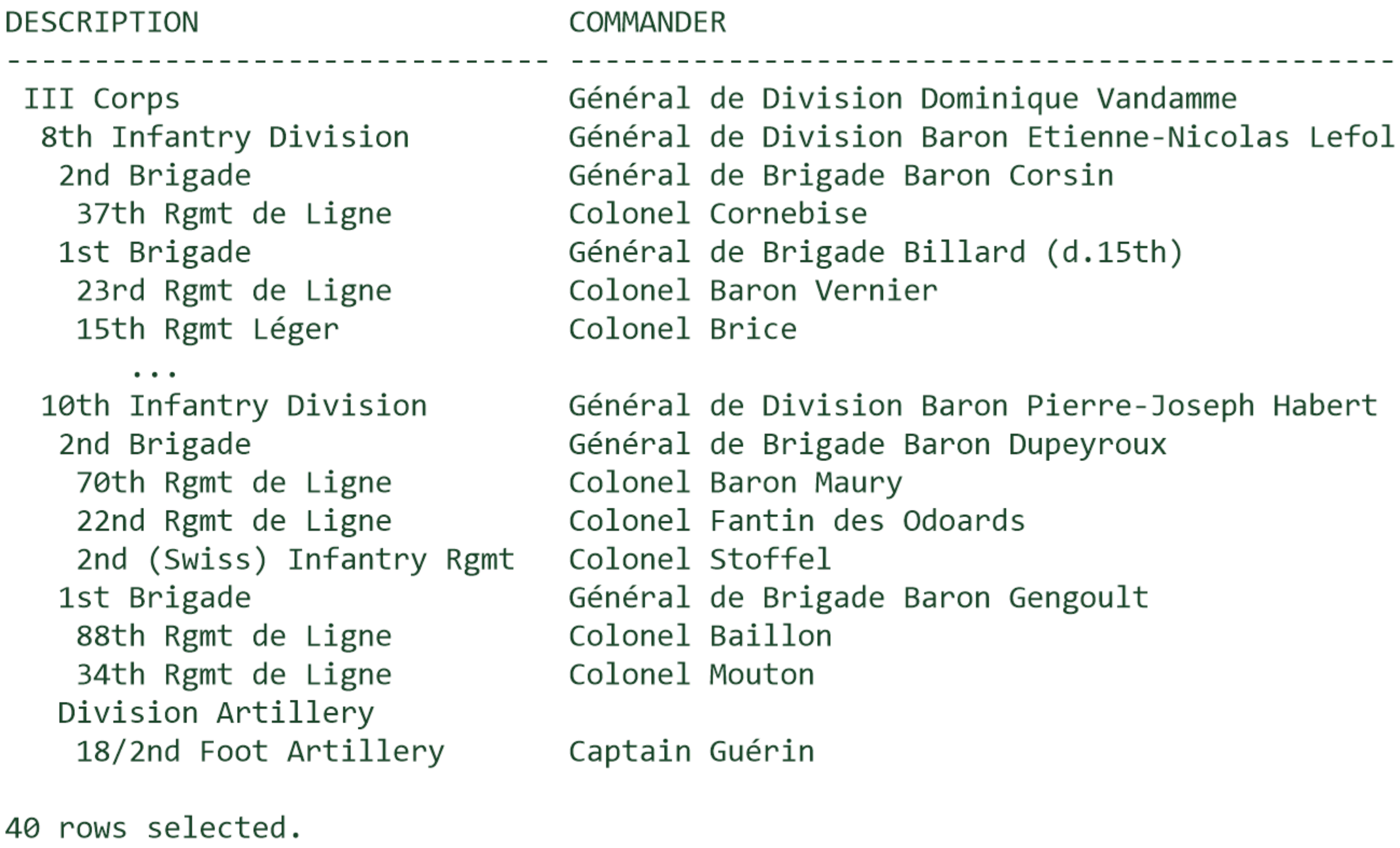
自顶向下查询：Vandamme查询

- 邻接模式
  - connect by <*a column of the current row*> = prior <*a column of the previous row*>,
  - connect by <*a column of the previous row*> = prior <*a column of the current row*>



```
select lpad(description, length(description) + level) description,
        commander
   from adjacency_model
   connect by parent_id = prior id
   start with commander = 'Général de Division Dominique Vandamme'
```

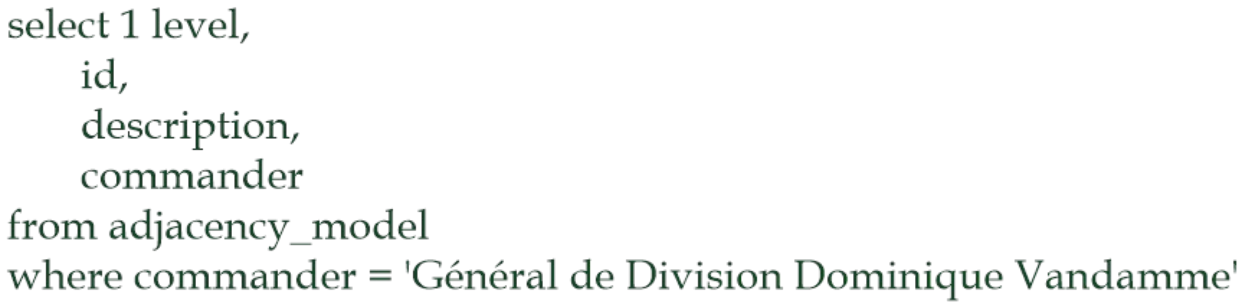
邻接模式



```
DESCRIPTION                     COMMANDER
------------------------------- -----------------------------------------------
 III Corps                      Général de Division Dominique Vandamme
  8th Infantry Division         Général de Division Baron Etienne-Nicolas Lefol
   2nd Brigade                  Général de Brigade Baron Corsin
    37th Rgmt de Ligne          Colonel Cornebise
   1st Brigade                  Général de Brigade Billard (d.15th)
    23rd Rgmt de Ligne          Colonel Baron Vernier
    15th Rgmt Léger             Colonel Brice
       ...
  10th Infantry Division        Général de Division Baron Pierre-Joseph Habert
   2nd Brigade                  Général de Brigade Baron Dupeyroux
    70th Rgmt de Ligne          Colonel Baron Maury
    22nd Rgmt de Ligne          Colonel Fantin des Odoards
    2nd (Swiss) Infantry Rgmt   Colonel Stoffel
   1st Brigade                  Général de Brigade Baron Gengoult
    88th Rgmt de Ligne          Colonel Baillon
    34th Rgmt de Ligne          Colonel Mouton
   Division Artillery
    18/2nd Foot Artillery       Captain Guérin

40 rows selected.
```

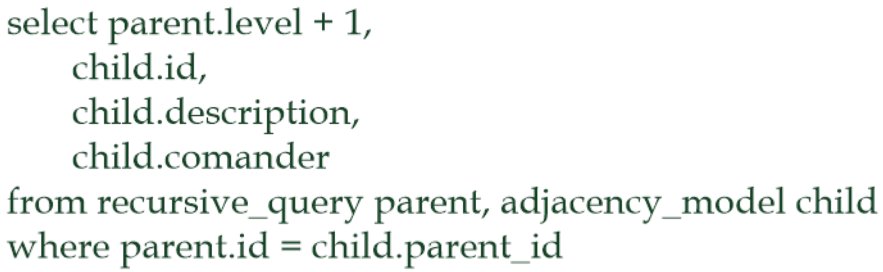
邻接模式:递归实现（MySQL8，CTE，common table expression）

- STEP 1： define starting point



```
select 1 level,
       id,
       description,
       commander
from adjacency_model
where commander = 'Général de Division Dominique Vandamme'
```

- STEP 2： define how each child row relates to its parent row



```
select parent.level + 1,
      child.id,
      child.description,
      child.comander
from recursive_query parent, adjacency_model child
where parent.id = child.parent_id
```

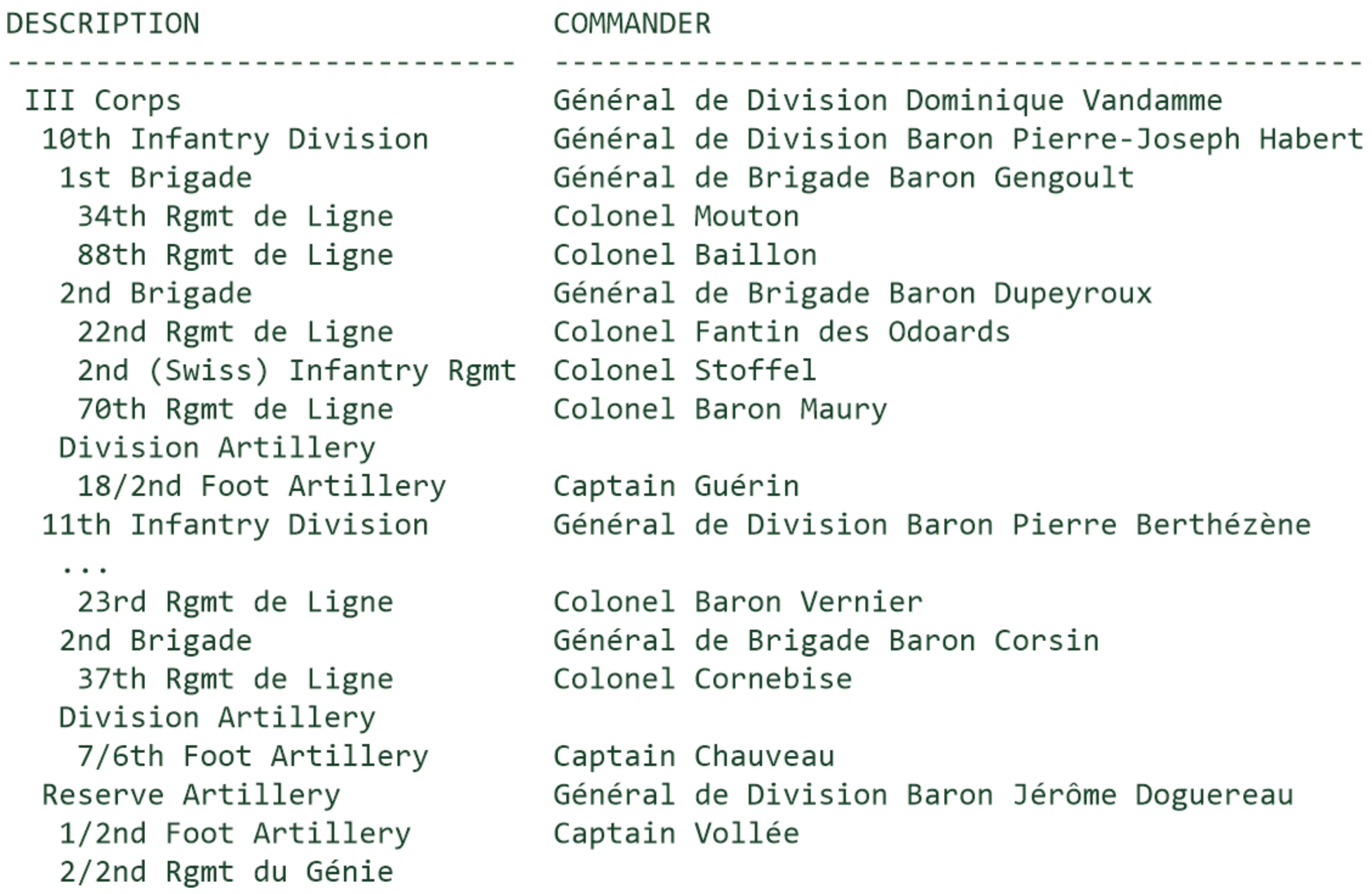
邻接模式:递归实现

```
with recursive_query(level, id, description, commander)
as (select 1 level,
        id,
        description,
        commander
   from adjacency_model
   where commander = 'Général de Division Dominique Vandamme'
   union all
   select parent.level + 1,
        child.id,
        child.description,
        child.commander
   from recursive_query parent,
       adjacency_model child
   where parent.id = child.parent_id)
select char(concat(repeat(' ', level), description), 60) description,
     commander
from recursive_query
```

邻接模式:递归实现

```
with recursive_query(level, id, rank, description, commander)
as (select 1,
           id,
           cast(1 as double),
           description,
           commander
     from adjacency_model
     where commander = 'Général de Division Dominique Vandamme'
     union all
     select parent.level + 1,
           child.id,
           parent.rank + ranking.sn / power(100.0, parent.level),
           child.description,
           child.commander
     from recursive_query parent,
         (select id,
                row_number( ) over (partition by parent_id
                                order by description) sn
          from adjacency_model) ranking,
         adjacency_model child
     where parent.id =child.parent_id
      and child.id = ranking.id)
select char(concat(repeat(' ', level), description), 60) description,
      commander
from recursive_query
order by rank
```

邻接模式:递归实现



```
DESCRIPTION                    COMMANDER
------------------------------ ----------------------------------------------
 III Corps                     Général de Division Dominique Vandamme
  10th Infantry Division       Général de Division Baron Pierre-Joseph Habert
   1st Brigade                 Général de Brigade Baron Gengoult
    34th Rgmt de Ligne         Colonel Mouton
    88th Rgmt de Ligne         Colonel Baillon
   2nd Brigade                 Général de Brigade Baron Dupeyroux
    22nd Rgmt de Ligne         Colonel Fantin des Odoards
    2nd (Swiss) Infantry Rgmt  Colonel Stoffel
    70th Rgmt de Ligne         Colonel Baron Maury
   Division Artillery
    18/2nd Foot Artillery      Captain Guérin
  11th Infantry Division       Général de Division Baron Pierre Berthézène
   ...
    23rd Rgmt de Ligne         Colonel Baron Vernier
   2nd Brigade                 Général de Brigade Baron Corsin
    37th Rgmt de Ligne         Colonel Cornebise
   Division Artillery
    7/6th Foot Artillery       Captain Chauveau
  Reserve Artillery            Général de Division Baron Jérôme Doguereau
   1/2nd Foot Artillery        Captain Vollée
   2/2nd Rgmt du Génie
```

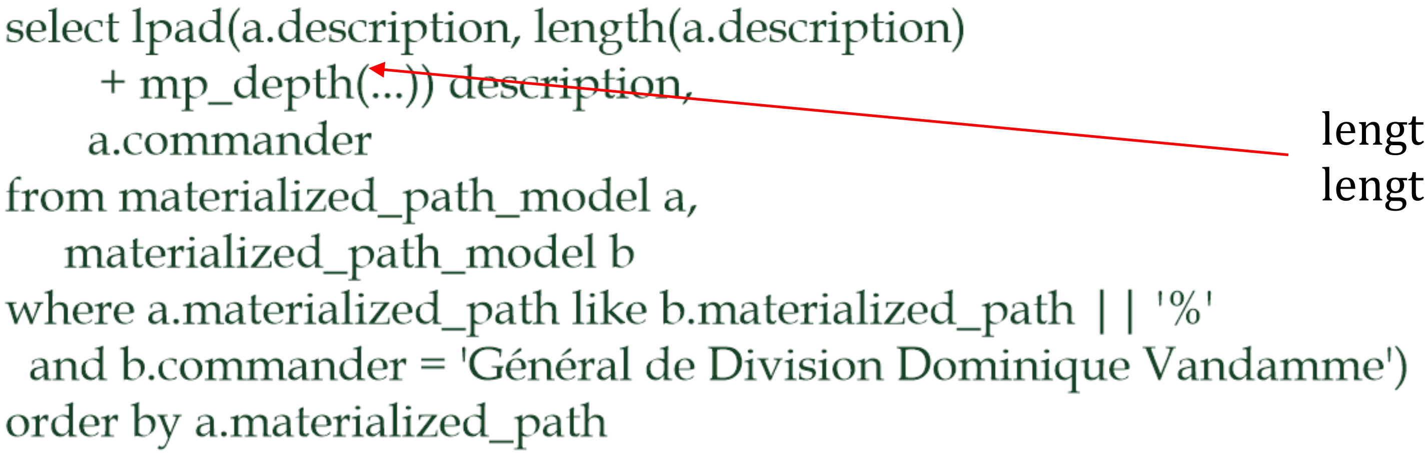
那 ……老的MySQL呢?

- 嗯……
- 两个方法
  - 手动union
  - 在一个查询中多次连接
  - 前提都是已知深度（自己眼睛看）

```
create view v1
as
select id, description, commander
  from adjacency_model
 where commander = 'Général de Division Dominique Vandamme’

create view v2
 as
select id, description, commander
  from adjacency_model
 where id =(select id from v1)

 create view v3
 as
select id, description, commander
  from adjacency_model
 where id =(select id from  v2)

select description, commander from v1
 union
select description, commander from v2
 union
select description, commander from v3
```

物化路径模型

- 查询编写不困难
- 计算由路径导出的层次不方便
- 假设mp_depth()函数返回当前节点深度

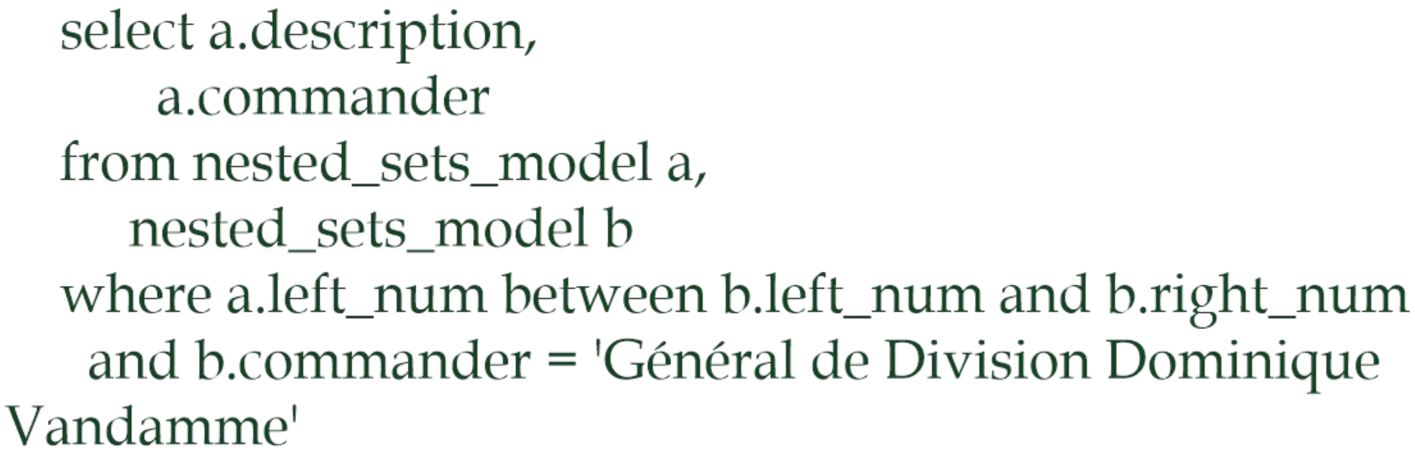


```
select lpad(a.description, length(a.description)
        + mp_depth(...)) description,
       a.commander
from materialized_path_model a,
     materialized_path_model b
where a.materialized_path like b.materialized_path || '%'
  and b.commander = 'Général de Division Dominique Vandamme')
order by a.materialized_path
```

length((materialized_path) -
length(replace(materialized_path, '.', ''))

嵌套集合模型

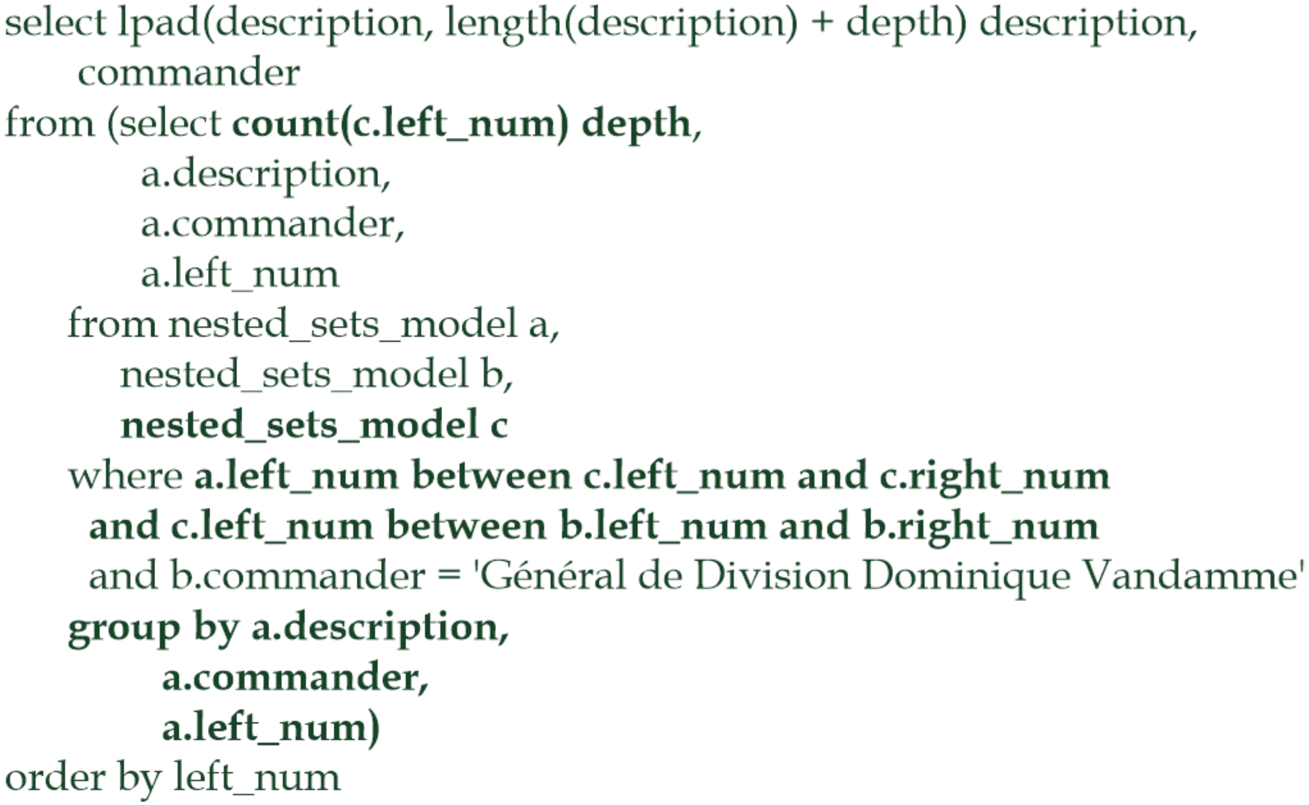
- 某节点的后代的left_num和right_num都会在该节点的left_num和right_num范围内



```
    select a.description,
           a.commander
    from nested_sets_model a,
         nested_sets_model b
    where a.left_num between b.left_num and b.right_num
      and b.commander = 'Général de Division Dominique
Vandamme'
```

嵌套集合模型

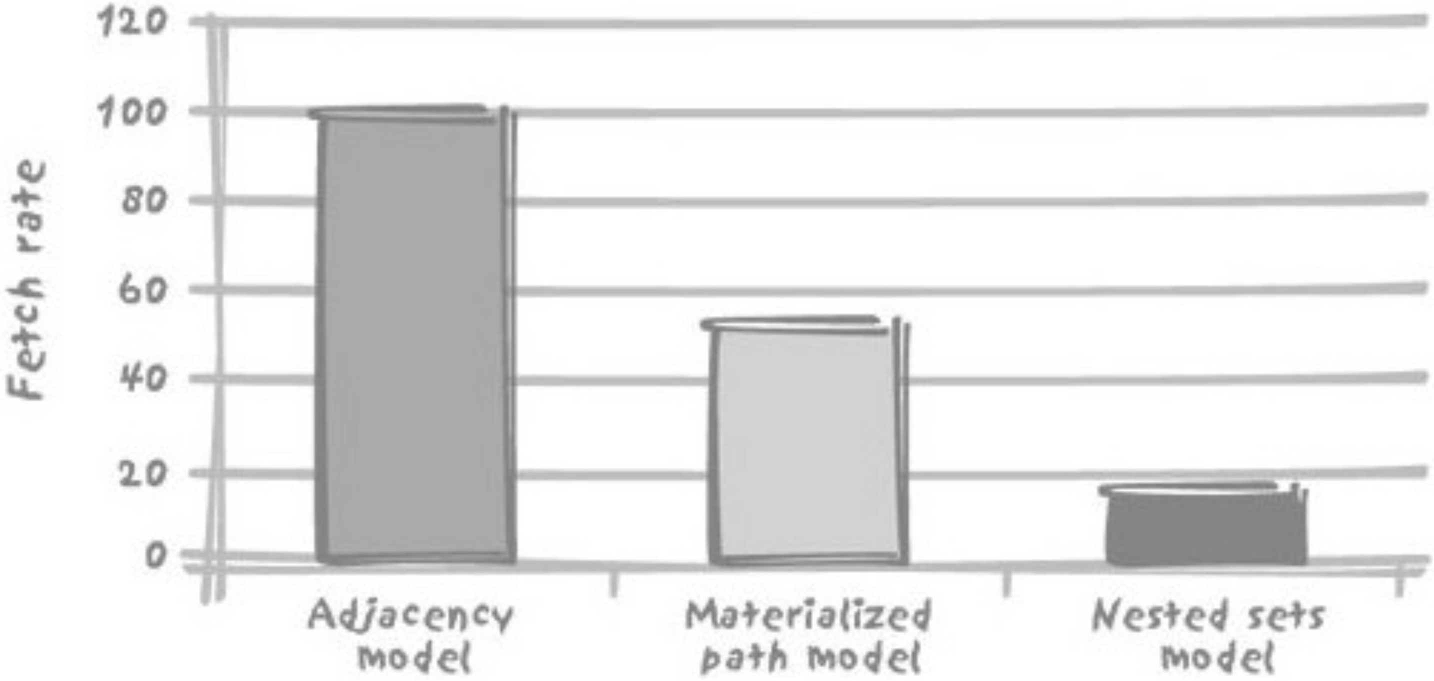
- 缩排怎么办……



```
select lpad(description, length(description) + depth) description,
       commander
from (select count(c.left_num) depth,
             a.description,
             a.commander,
             a.left_num
      from nested_sets_model a,
           nested_sets_model b,
           nested_sets_model c
      where a.left_num between c.left_num and c.right_num
        and c.left_num between b.left_num and b.right_num
        and b.commander = 'Général de Division Dominique Vandamme'
      group by a.description,
               a.commander,
               a.left_num)
order by left_num
```

比较各模型下的Vandamme模型

- 返回40条记录，循环执行每个查询5000次，比较每秒返回的记录数



递归 SQL 的语法

```
WITH RECURSIVE cte_name (column_list) AS (
  -- 初始查询
  SELECT ...
  UNION ALL
  -- 递归查询
  SELECT ...
  FROM cte_name
  WHERE ...
)
-- 主查询
SELECT ...
FROM cte_name
WHERE ...;
```

- WITH RECURSIVE：定义递归公共表表达式的开始。
- cte_name：递归公共表表达式的名称。
- column_list：列出递归公共表中包含的列。
- SELECT：初始查询部分，用于指定初始结果集。
- UNION ALL：连接初始查询和递归查询的操作符。
- 第二个 SELECT：递归查询部分，定义了如何从已有结果集中生成新的结果集。
- FROM cte_name：在递归查询中引用递归公共表自身。
- WHERE：可选的过滤条件，用于限制递归的终止条件。
- 主查询：可以在递归公共表表达式之后进行查询，也可以在其他查询中使用递归公共表表达式。

递归 SQL 的层级引入

```
WITH RECURSIVE DepartmentHierarchy AS (
    -- 初始查询：选择顶级部门（没有父部门）作为起点
    SELECT dept_id, parent_dept_id, dept_name, 1 AS level
    FROM Departments
    WHERE parent_dept_id IS NULL

    UNION ALL

    -- 递归查询：连接上一级部门和当前部门
    SELECT d.dept_id, d.parent_dept_id, d.dept_name, dh.level + 1
    FROM Departments d
    JOIN DepartmentHierarchy dh ON d.parent_dept_id = dh.dept_id
)
    -- 主查询：查询所有部门及其层级关系
SELECT dept_id, parent_dept_id, dept_name, level
FROM DepartmentHierarchy
ORDER BY level, dept_id;
```

假设有一个部门表（Departments），其中包含部门ID（dept_id）和父部门ID（parent_dept_id）两列，用于表示部门之间的层级关系。我们希望使用递归CTE来查询每个部门及其所有子部门的层级关系。

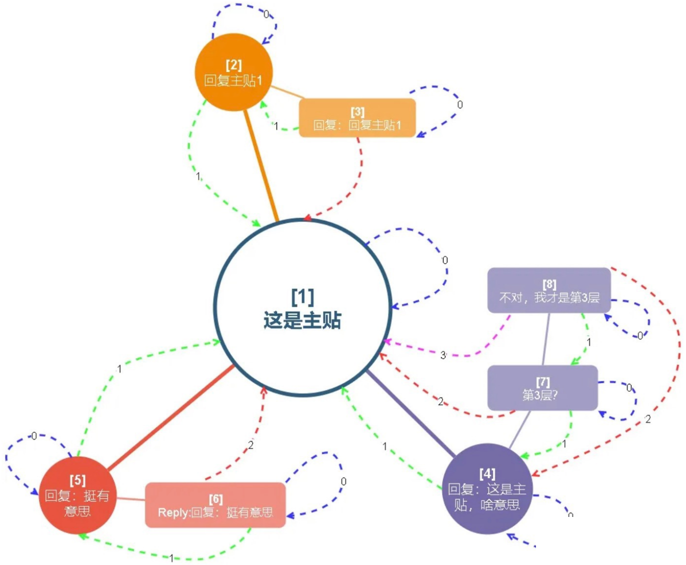
有没有其他设计模型？闭包表模型 (closure table model)

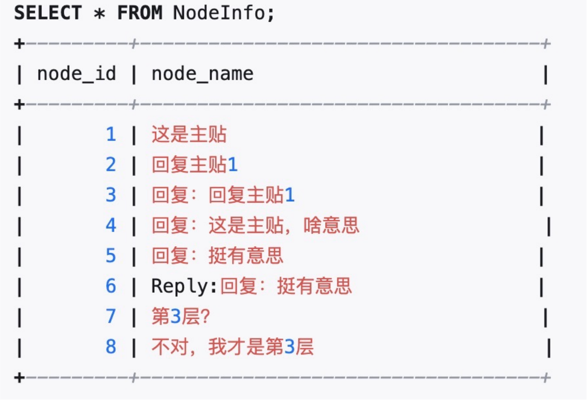
```
CREATE TABLE `NodeInfo` (
         `node_id` INT NOT NULL AUTO_INCREMENT,
        `node_name` VARCHAR (255),
        PRIMARY KEY (`node_id`)
) DEFAULT CHARSET = utf8mb4;

CREATE TABLE `NodeRelation` (
         `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '自增ID’,
        `ancestor` INT(10) UNSIGNED NOT NULL DEFAULT '0' COMMENT '祖先节点’,
         `descendant` INT(10) UNSIGNED NOT NULL DEFAULT '0' COMMENT '后代节点’,
        `distance` TINYINT(3) UNSIGNED NOT NULL DEFAULT '0' COMMENT '相隔层级，>=1’,
        PRIMARY KEY (`id`),
        UNIQUE KEY `uniq_anc_desc` (`ancestor`,`descendant`),
        KEY `idx_desc` (`descendant`)
) ENGINE = InnoDB DEFAULT CHARSET = utf8mb4 COMMENT = '节点关系表'
```

如何防止数据出错

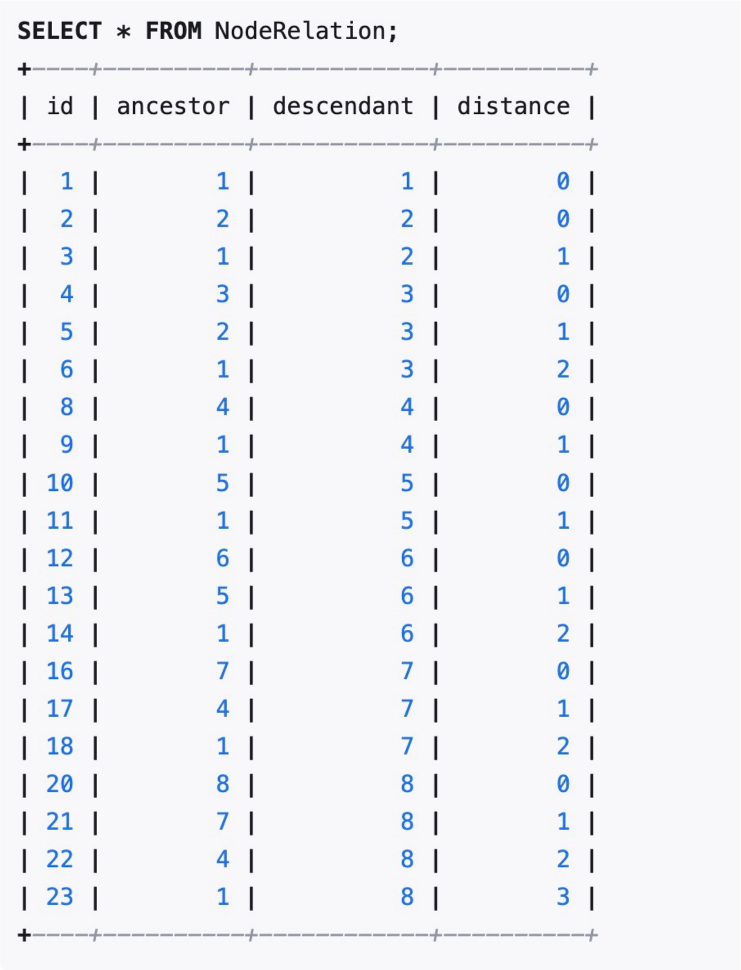
```
CREATE DEFINER = `root`@`localhost` PROCEDURE `AddNode`(`_parent_name` varchar(255),`_node_name` varchar(255))
BEGIN
        DECLARE _ancestor INT(10) UNSIGNED;
        DECLARE _descendant INT(10) UNSIGNED;
         DECLARE _parent INT(10) UNSIGNED;
        IF NOT EXISTS(SELECT node_id From NodeInfo WHERE node_name = _node_name)
        THEN
                INSERT INTO NodeInfo (node_name) VALUES(_node_name);
                SET _descendant = (SELECT node_id FROM NodeInfo WHERE node_name = _node_name);
                INSERT INTO NodeRelation (ancestor,descendant,distance) VALUES(_descendant,_descendant,0);

                 IF EXISTS (SELECT node_id FROM NodeInfo WHERE node_name = _parent_name)
                THEN
                        SET _parent = (SELECT node_id FROM NodeInfo WHERE node_name = _parent_name);
                        INSERT INTO NodeRelation (ancestor,descendant,distance)
                                SELECT ancestor,_descendant,distance+1 FROM NodeRelation WHERE descendant = _parent;
                END IF;
        END IF;
END;
```

Sample：论坛回帖





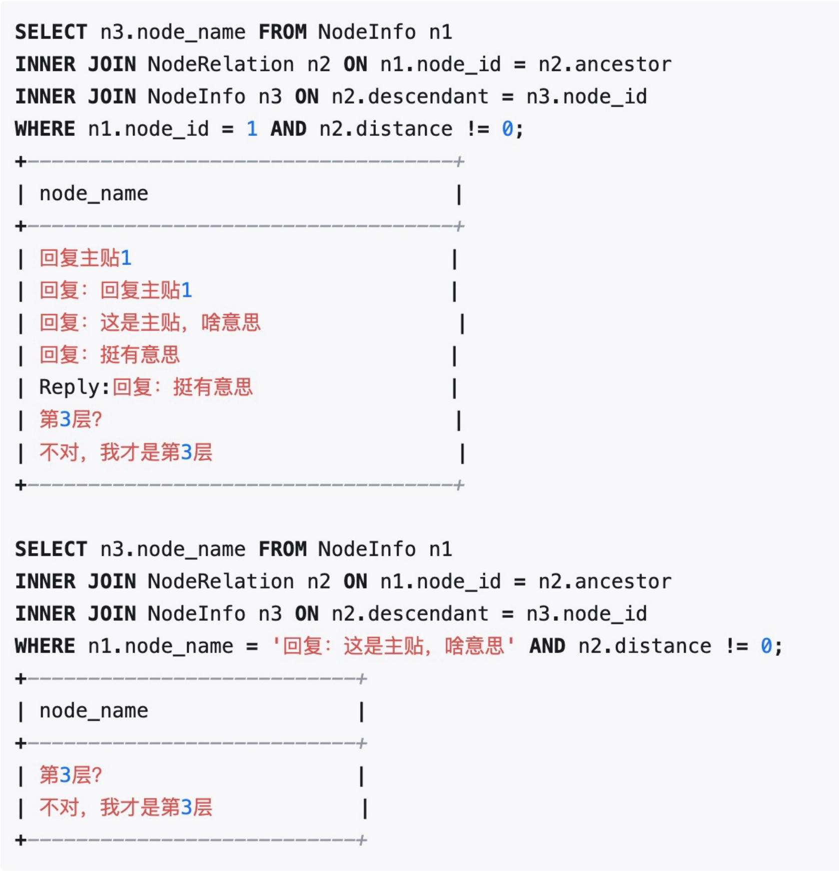
```
SELECT * FROM NodeInfo;
+---------+------------------------------------+
| node_id | node_name                          |
+---------+------------------------------------+
|       1 | 这是主贴                            |
|       2 | 回复主贴1                           |
|       3 | 回复：回复主贴1                      |
|       4 | 回复：这是主贴，啥意思                 |
|       5 | 回复：挺有意思                       |
|       6 | Reply:回复：挺有意思                 |
|       7 | 第3层?                              |
|       8 | 不对，我才是第3层                     |
+---------+------------------------------------+
```



```
SELECT * FROM NodeRelation;
+----+----------+------------+----------+
| id | ancestor | descendant | distance |
+----+----------+------------+----------+
|  1 |        1 |          1 |        0 |
|  2 |        2 |          2 |        0 |
|  3 |        1 |          2 |        1 |
|  4 |        3 |          3 |        0 |
|  5 |        2 |          3 |        1 |
|  6 |        1 |          3 |        2 |
|  8 |        4 |          4 |        0 |
|  9 |        1 |          4 |        1 |
| 10 |        5 |          5 |        0 |
| 11 |        1 |          5 |        1 |
| 12 |        6 |          6 |        0 |
| 13 |        5 |          6 |        1 |
| 14 |        1 |          6 |        2 |
| 16 |        7 |          7 |        0 |
| 17 |        4 |          7 |        1 |
| 18 |        1 |          7 |        2 |
| 20 |        8 |          8 |        0 |
| 21 |        7 |          8 |        1 |
| 22 |        4 |          8 |        2 |
| 23 |        1 |          8 |        3 |
+----+----------+------------+----------+
```

Sample：论坛回帖查询

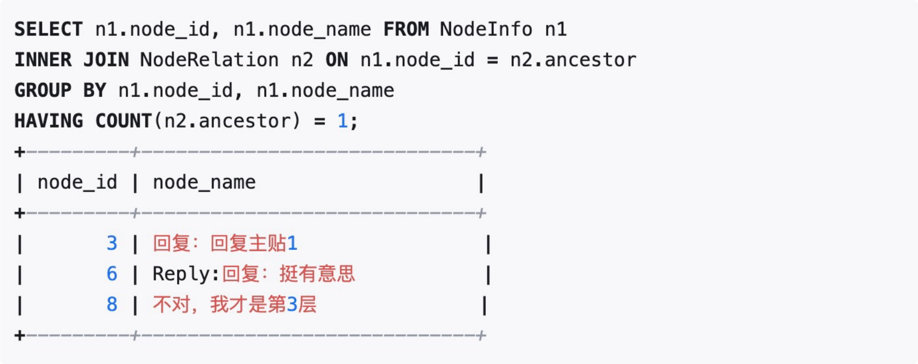
获取闭包表全树或子树



```
SELECT n3.node_name FROM NodeInfo n1
INNER JOIN NodeRelation n2 ON n1.node_id = n2.ancestor
INNER JOIN NodeInfo n3 ON n2.descendant = n3.node_id
WHERE n1.node_id = 1 AND n2.distance != 0;
+-------------------------------------+
| node_name                           |
+-------------------------------------+
| 回复主贴1                            |
| 回复：回复主贴1                       |
| 回复：这是主贴，啥意思                  |
| 回复：挺有意思                        |
| Reply:回复：挺有意思                  |
| 第3层?                               |
| 不对，我才是第3层                      |
+-------------------------------------+

SELECT n3.node_name FROM NodeInfo n1
INNER JOIN NodeRelation n2 ON n1.node_id = n2.ancestor
INNER JOIN NodeInfo n3 ON n2.descendant = n3.node_id
WHERE n1.node_name = '回复：这是主贴，啥意思' AND n2.distance != 0;
+-----------------------------+
| node_name                   |
+-----------------------------+
| 第3层?                       |
| 不对，我才是第3层              |
+-----------------------------+
```

通过关联表的父子关系，去掉自指的记录，使用内连接获取所有子节点。

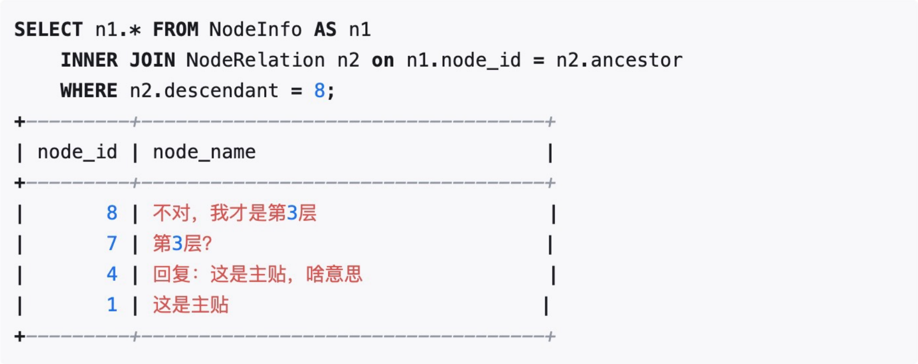
获取闭包表叶节点



```
SELECT n1.node_id, n1.node_name FROM NodeInfo n1
INNER JOIN NodeRelation n2 ON n1.node_id = n2.ancestor
GROUP BY n1.node_id, n1.node_name
HAVING COUNT(n2.ancestor) = 1;
+---------+-----------------------------+
| node_id | node_name                   |
+---------+-----------------------------+
|       3 | 回复：回复主贴1               |
|       6 | Reply:回复：挺有意思          |
|       8 | 不对，我才是第3层              |
+---------+-----------------------------+
```

叶节点的特征是没有子节点，所以它的 ID 只会在关联表的 ancestor 字段出现一次，就是自指的那一次。

获取闭包表父节点



```
SELECT n1.* FROM NodeInfo AS n1
    INNER JOIN NodeRelation n2 on n1.node_id = n2.ancestor
    WHERE n2.descendant = 8;
+---------+-----------------------------------+
| node_id | node_name                         |
+---------+-----------------------------------+
|       8 | 不对，我才是第3层                    |
|       7 | 第3层?                             |
|       4 | 回复：这是主贴，啥意思                |
|       1 | 这是主贴                            |
+---------+-----------------------------------+
```

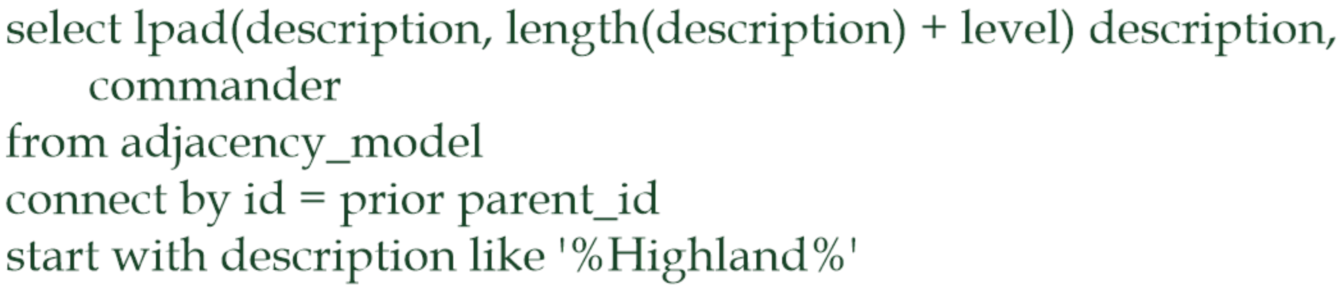
从关系表来倒查，因为关系表里每个节点与其所有上级的关系都记录了。

自底向上访问：Highland查询

- 在description字段中查找“Highland”字符串
- 必然导致完整的表扫描
- 不同模型下Highland查询的差异

邻接模式

- Connect by相当容易实现

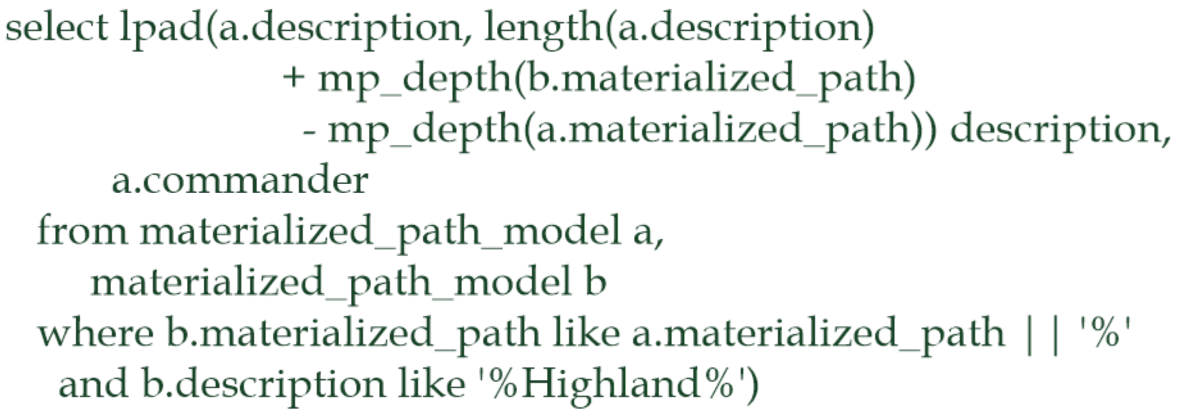


```
select lpad(description, length(description) + level) description,
      commander
from adjacency_model
connect by id = prior parent_id
start with description like '%Highland%'
```

```
DESCRIPTION                        COMMANDER
  ---------------------------------- ----------------------------------------
   2/73rd (Highland) Rgmt of Foot    Lt-Colonel William George Harris
    5th British Brigade              Major-General Sir Colin Halkett
     3rd Anglo-German Division       Lt-General Count Charles von Alten
      I Corps                        Prince William of Orange
       The Anglo-Allied Army of 1815 Field Marshal Arthur Wellesley, Duke of
                                     Wellington
   1/71st (Highland) Rgmt of Foot    Lt-Colonel Thomas Reynell
    British Light Brigade            Major-General Frederick Adam
     2nd Anglo-German Division       Lt-General Sir Henry Clinton
      II Corps                       Lieutenant-General Lord Rowland Hill
       The Anglo-Allied Army of 1815 Field Marshal Arthur Wellesley, Duke of
                                     Wellington
   1/79th (Highland) Rgmt of Foot    Lt-Colonel Neil Douglas
    8th British Brigade              Lt-General Sir James Kempt
     5th Anglo-German Division       Lt-General Sir Thomas Picton (d.18th)
      General Reserve                Duke of Wellington
       The Anglo-Allied Army of 1815 Field Marshal Arthur Wellesley, Duke of
                                     Wellington
   1/42nd (Highland) Rgmt of Foot    Colonel Sir Robert Macara (d.16th)
    9th British Brigade              Major-General Sir Denis Pack
     5th Anglo-German Division       Lt-General Sir Thomas Picton (d.18th)
      General Reserve                Duke of Wellington
       The Anglo-Allied Army of 1815 Field Marshal Arthur Wellesley, Duke of
                                     Wellington
   1/92nd (Highland) Rgmt of Foot    Lt-Colonel John Cameron
    9th British Brigade              Major-General Sir Denis Pack
     5th Anglo-German Division       Lt-General Sir Thomas Picton (d.18th)
      General Reserve                Duke of Wellington
       The Anglo-Allied Army of 1815 Field Marshal Arthur Wellesley, Duke of
                                     Wellington

  25 rows selected.
```

物化路径模型

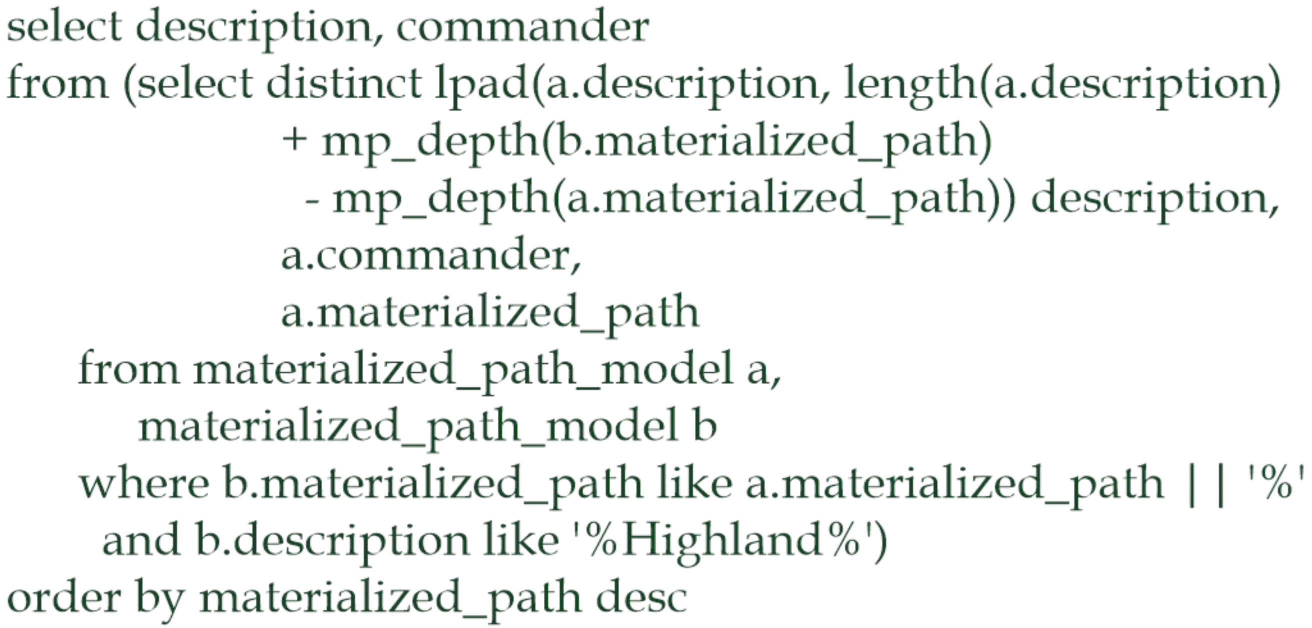
- 仅找出适当的记录并缩排显示算容易



```
select lpad(a.description, length(a.description)
                        + mp_depth(b.materialized_path)
                         - mp_depth(a.materialized_path)) description,
        a.commander
   from materialized_path_model a,
        materialized_path_model b
   where b.materialized_path like a.materialized_path || '%'
    and b.description like '%Highland%')
```

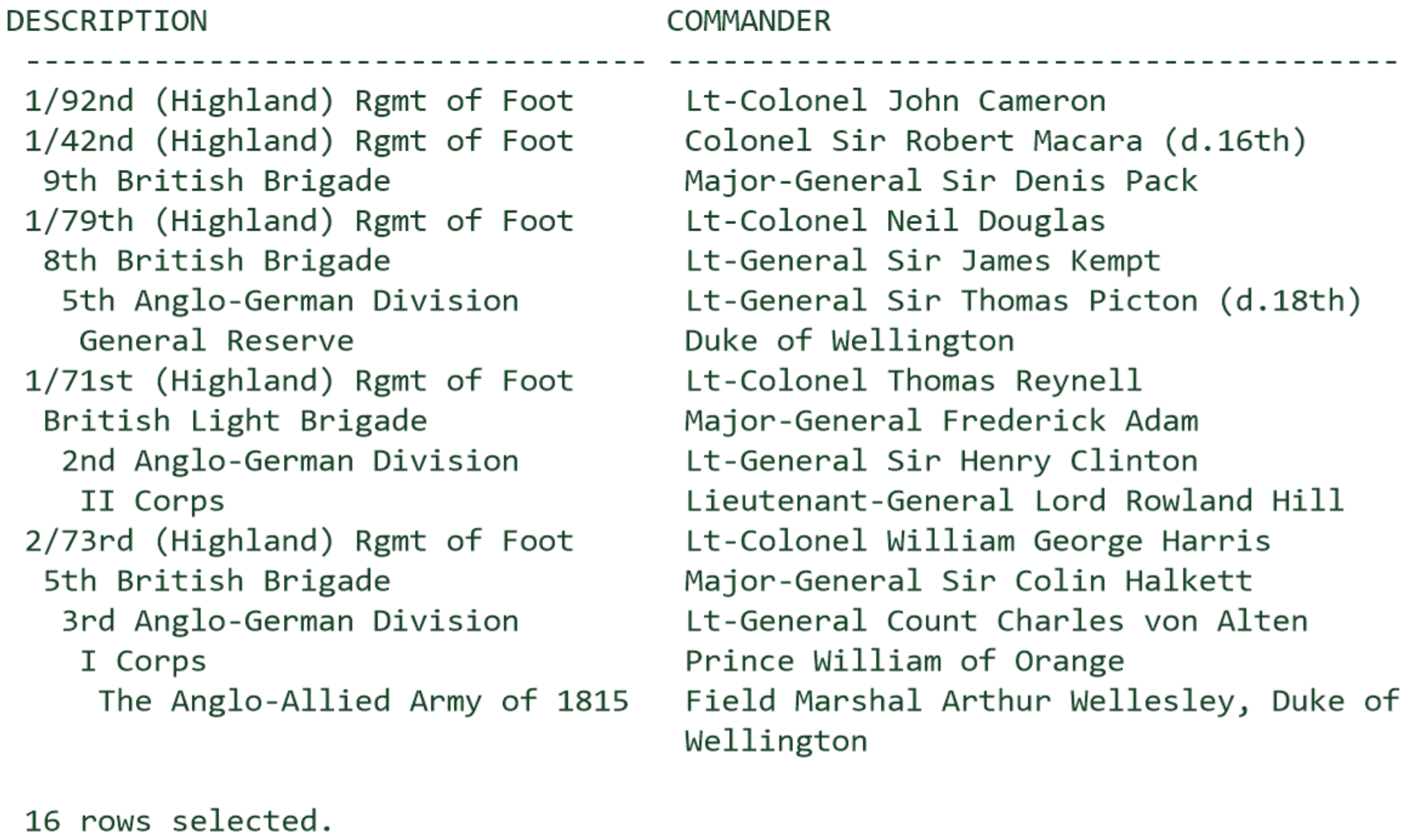
  - 重复记录的问题
  - 顺序的问题

物化路径模型



```
select description, commander
from (select distinct lpad(a.description, length(a.description)
                    + mp_depth(b.materialized_path)
                     - mp_depth(a.materialized_path)) description,
                    a.commander,
                    a.materialized_path
      from materialized_path_model a,
           materialized_path_model b
      where b.materialized_path like a.materialized_path || '%'
       and b.description like '%Highland%')
order by materialized_path desc
```

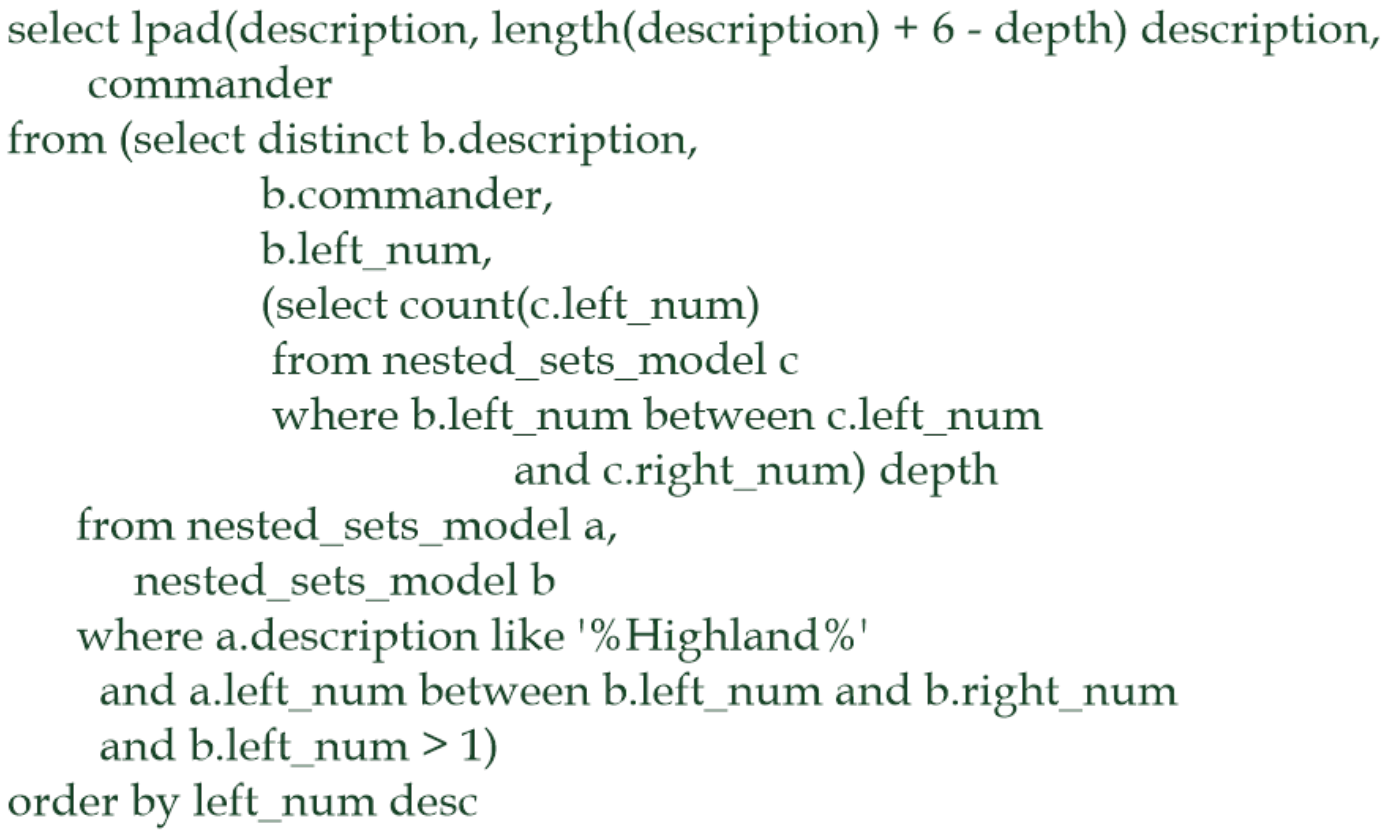
much nicer and more compact result



```
DESCRIPTION                         COMMANDER
 ---------------------------------- ----------------------------------------
 1/92nd (Highland) Rgmt of Foot      Lt-Colonel John Cameron
 1/42nd (Highland) Rgmt of Foot      Colonel Sir Robert Macara (d.16th)
  9th British Brigade                Major-General Sir Denis Pack
 1/79th (Highland) Rgmt of Foot      Lt-Colonel Neil Douglas
  8th British Brigade                Lt-General Sir James Kempt
   5th Anglo-German Division         Lt-General Sir Thomas Picton (d.18th)
    General Reserve                  Duke of Wellington
 1/71st (Highland) Rgmt of Foot      Lt-Colonel Thomas Reynell
  British Light Brigade              Major-General Frederick Adam
   2nd Anglo-German Division         Lt-General Sir Henry Clinton
    II Corps                         Lieutenant-General Lord Rowland Hill
 2/73rd (Highland) Rgmt of Foot      Lt-Colonel William George Harris
  5th British Brigade                Major-General Sir Colin Halkett
   3rd Anglo-German Division         Lt-General Count Charles von Alten
    I Corps                          Prince William of Orange
     The Anglo-Allied Army of 1815   Field Marshal Arthur Wellesley, Duke of
                                     Wellington

 16 rows selected.
```

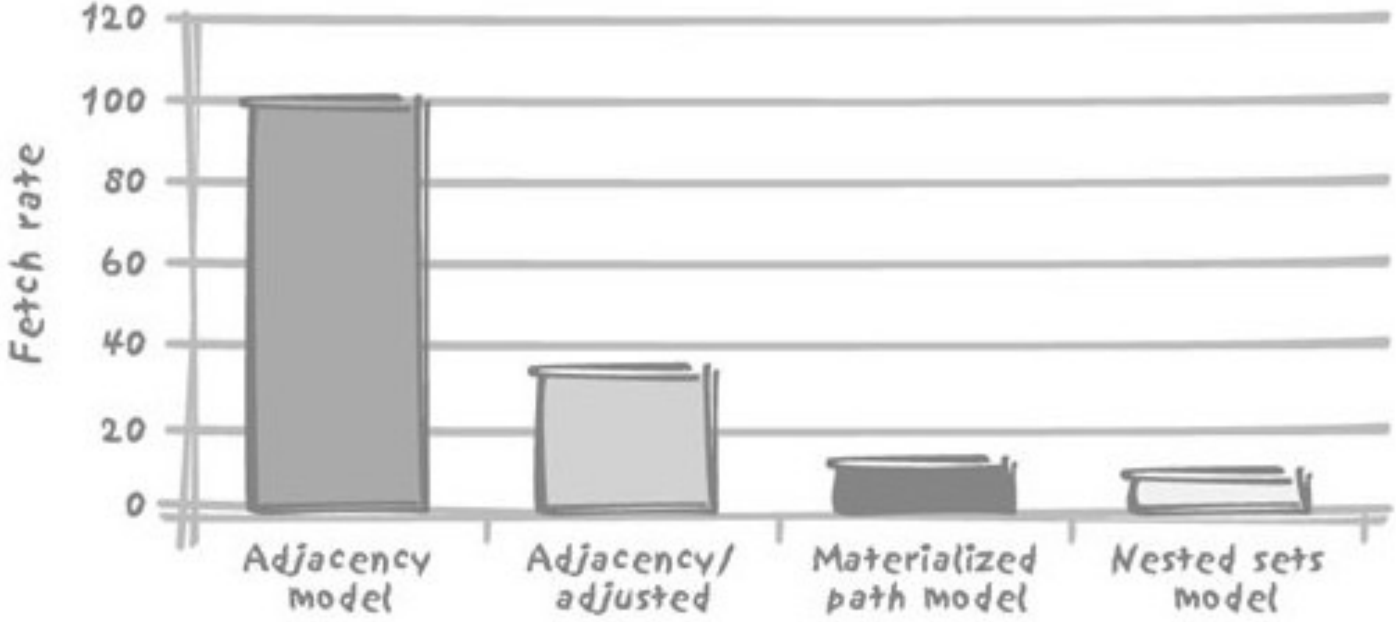
嵌套集合模型

- 动态计算深度依旧是个问题
- 不要显示人造根节点
- 硬编码最大深度（为了缩排显示）



```
select lpad(description, length(description) + 6 - depth) description,
       commander
from (select distinct b.description,
                      b.commander,
                      b.left_num,
                      (select count(c.left_num)
                       from nested_sets_model c
                       where b.left_num between c.left_num
                                            and c.right_num) depth
      from nested_sets_model a,
           nested_sets_model b
      where a.description like '%Highland%'
        and a.left_num between b.left_num and b.right_num
        and b.left_num > 1)
order by left_num desc
```

比较各种模型下的Highland查询



一些问题

- 物化路径不该是KEY，即使他们有唯一性
- 物化路径和邻接模型等价使用的时候，不该暗示任何兄弟节点的排序
- 所选择的编码方式不需要完全中立

到底哪种模型效率更高?

- 邻接模式/父子关系模型
    - 简单，成熟，深度是最大的障碍
- 物化路径模型/路径枚举模型
    - 读取和修改的平衡，稳定的输出
- 嵌套集合模型
    - 读取频率高于修改频率，只在乎上下关系，不在乎层级（他是谁的人，他的人有谁）
- 闭包表模型
    - 额外表的存储，维护细节和成本升高，但查询效率优

Practice in class 6-1

- 课程中的例子使用了oracle，请尝试使用MySQL写成三种模型下的自顶向下和自底向上的两种查询模式的查询（共6个查询）
- 对邻接模型，查询会非常繁琐，你体会一下会提高你的SQL能力，特别是如何进行缩排。

对保存于叶节点的值做聚合

表UNITS

```
ID NAME                       COMMANDER
-- -------------------------- -----------------------------------------------
 1 III Corps                  Général de Division Dominique Vandamme
 2 8th Infantry Division      Général de Division Baron Etienne-Nicolas Lefol
 3 1st Brigade                Général de Brigade Billard
 4 2nd Brigade                Général de Brigade Baron Corsin
 5 10th Infantry Division     Général de Division Baron Pierre-Joseph Habert
 6 1st Brigade                Général de Brigade Baron Gengoult
 7 2nd Brigade                Général de Brigade Baron Dupeyroux
 8 11th Infantry Division     Général de Division Baron Pierre Berthézène
 9 1st Brigade                Général de Brigade Baron Dufour
10 2nd Brigade                Général de Brigade Baron Logarde
11 3rd Light Cavalry Division Général de Division Baron Jean-Simon Domont
12 1st Brigade                Général de Brigade Baron Dommanget
13 2nd Brigade                Général de Brigade Baron Vinot
14 Reserve Artillery          Général de Division Baron Jérôme Doguereau
```

UNIT_LINKS_ADJACENCY

ID	PARENT_ID
2	1
3	2
4	2
5	1
6	5
7	5
8	1
9	8
10	8
11	1
12	11
13	11
14	1

UNIT_LINKS_PATH

ID	PATH
1	1
2	1.1
3	1.1.1
4	1.1.2
5	1.2
6	1.2.1
7	1.2.2
8	1.3
9	1.3.1
10	1.3.2
11	1.4
12	1.4.1
13	1.4.2
14	1.5

UNIT_STRENGTH

ID	MEN
3	2952
4	2107
6	2761
7	2823
9	2488
10	2050
12	699
13	318
14	152

计算每一层的人数（邻接模型）

计算第三军的总人数：

```
select sum(men)
   from unit_strength
   where id in (select id
             from unit_links_adjacency
             connect by prior id = parent_id
             start with parent_id = 1)
```

Connect by 的过程化本质带来巨大的障碍

计算每一层的人数

```
select u.name,
    u.commander,
    (select sum(men)
     from unit_strength
     where id in (select id
            from unit_links_adjacency
            connect by parent_id = prior id
            start with parent_id = u.id)
      or id = u.id) men
from units u
```

计算每一层的人数（物化路径）

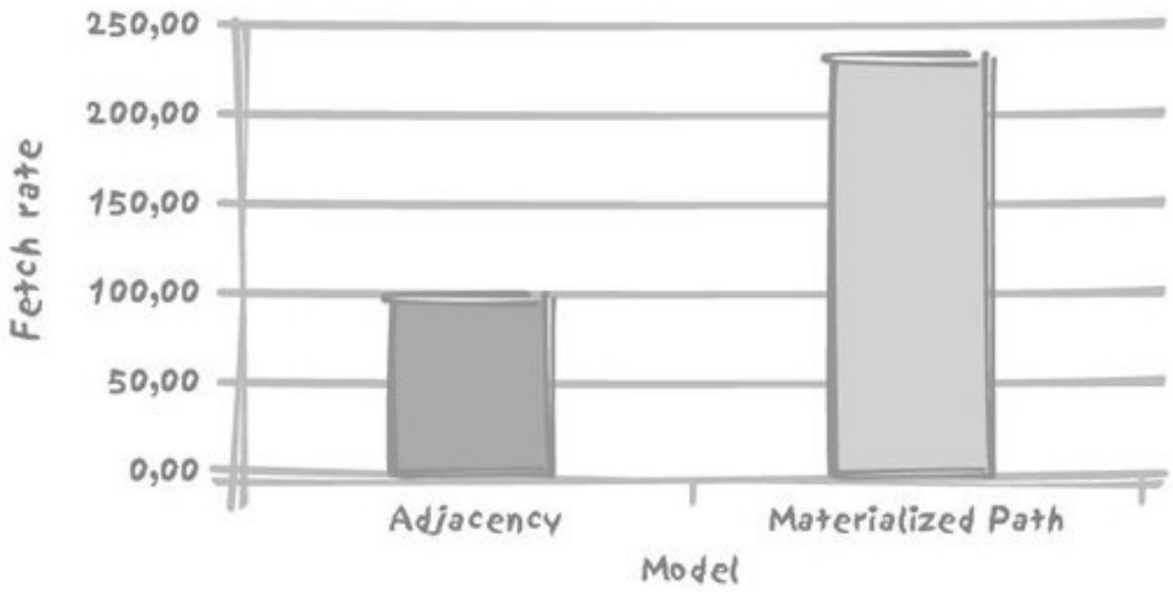
```
SQL> select * from exploded_links_path;
        ID   ANCESTOR      DEPTH
---------- ---------- ----------
        14          1          1
        13          1          2
        12          1          2
        11          1          1
        10          1          2
         9          1          2
         8          1          1
         7          1          2
         6          1          2
         5          1          1
         4          1          2
         3          1          2
         2          1          1
         4          2          1
         3          2          1
         7          5          1
         6          5          1
        10          8          1
         9          8          1
        13         11          1
        12         11          1
```

```
select u.name, u.commander, sum(s.men) men
  from units u,
       exploded_links_path el,
       unit_strength s
  where u.id = el.ancestor
   and el.id = s.id
  group by u.name, u.commander
```

```
NAME                           COMMANDER                                     MEN
----------------------         ------------------------------------          -----
III Corps                      Général de Division Dominique Vandamme        16350
8th Infantry Division          Général de Division Baron Etienne-            5059
                               Nicolas Lefol
10th Infantry Division         Général de Division Baron Pierre              5584
                               Joseph Habert
11th Infantry Division         Général de Division Baron Pierre              4538
                               Berthézène
3rd Light Cavalry Division     Général de Division Baron Jean-Simon          1017
                               Domont
```

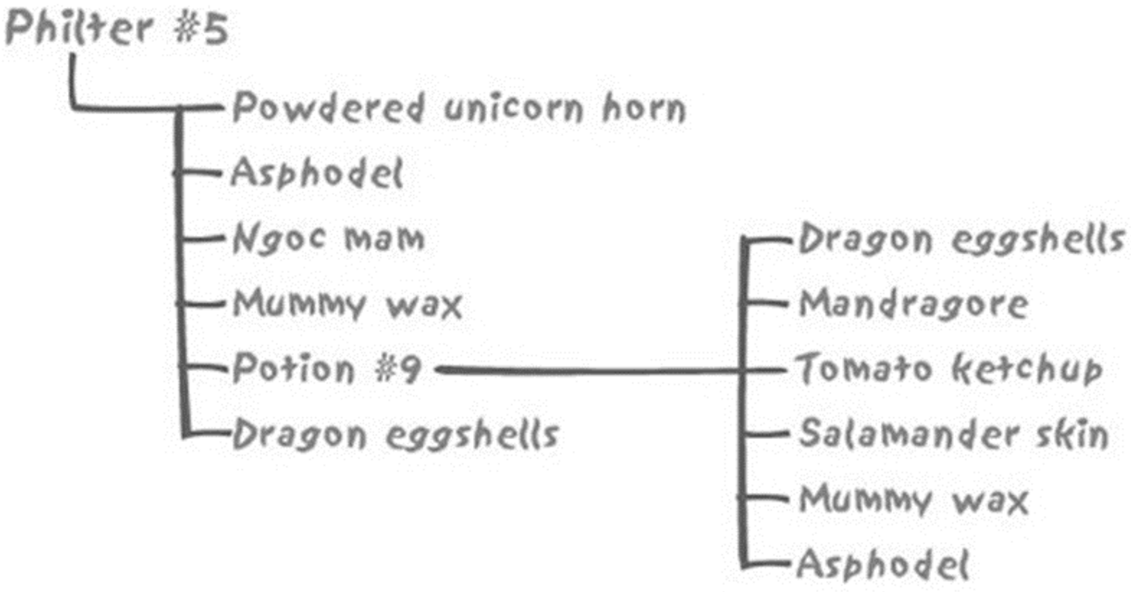
计算每一层的人数

- 执行查询5000次，比较单位时间返回的记录数



散布在各层的百分比

- 每种魔药由多种成分（ingredient）组成，处方（recipe）列出成分及百分比。处方可以共享某种“基础魔药”，以复合成分（compound ingredient）的形式表示。



多叉树（Multiway Tree)

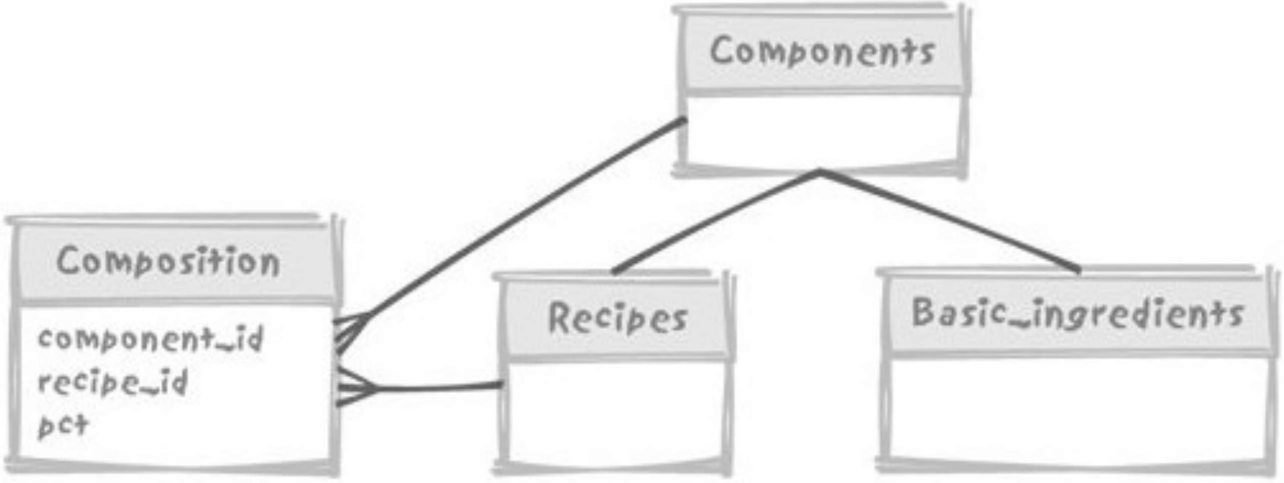
- 复杂的层级关系
- 每个节点有多父节点，每个节点也有多个子节点
- 边列表结构库设计（Edge List），这也是图结构的基础数据库设计
- 这也是闭包模型的扩展——具体应用中，可能需要增加一些实体

```
CREATE TABLE Node (
    node_id INT PRIMARY KEY,
    node_name VARCHAR(255)
);
```

```
CREATE TABLE EdgeList (
    edge_id INT PRIMARY KEY,
    parent_id INT,
    child_id INT,
    FOREIGN KEY (parent_id) REFERENCES Node(node_id),
    FOREIGN KEY (child_id) REFERENCES Node(node_id)
);
```

散布在各层的百分比

- 某一种可以选择的建模方法
- Components表为通用类型
- 它有recipes和basic_ingredients两种子类型
- Composition表保存处方成分（可以是处方或基本成分及其数量）



散布在各层的百分比

```
SQL> select connect_by_root recipe_id root_recipe,
  2         recipe_id,
  3         prior pct,
  4         pct
  5         component_id
  6  from composition
  7  connect by recipe_id = prior component_id
  8  /
```

```
ROOT_RECIPE   RECIPE_ID  PRIORPCT   PCT        COMPONENT_ID
-----------   ---------- ---------- ---------  ------------
         14   14                    5          3
         14   14                    20         7
         14   14                    15         8
         14   14                    30         9
         14   14                    20         10
         14   14                    10         2
         15   15                    30         14
         15   14         30         5          3
         15   14         30         20         7
         15   14         30         15         8
         15   14         30         30         9
...
```

```
with recursive_composition(actual_pct, component_id)
  as (select a.pct,
         a.component_id
    from composition a,
       components b
    where b.component_id = a.recipe_id
     and b.component_name = 'Philter #5'
    union all
    select parent.pct * child.pct,
         child.component_id
    from recursive_composition parent,
       composition child
    where child.recipe_id = parent.component_id)


 select x.component_name, sum(y.actual_pct)
   from recursive_composition y,
        components x
   where x.component_id = y.component_id
      and x.component_type = 'I'
   group by x.component_name
```

树状结构的问题

- 本章的方法，在数据量很少的情况下效果令人满意
- 对大数据量的处理“像老爷车一样慢”

- 同样可以采用非规范化模型、或基于触发器的扁平化数据模型。
- 不建议对关系模型“屡遭诟病的缓慢本性”反规范化，这很容易遮掩程序设计中的问题。
- 不过，SQL确实缺乏处理树结构的强大的、可伸缩的手段。

End

下一讲，进入SQL 的部分