关系数据库存储树形结构数据的理想实践

王先明 2010.02

1
开篇思考

2
主流方案

3
改进方案

1 开篇思考

树形结构数据

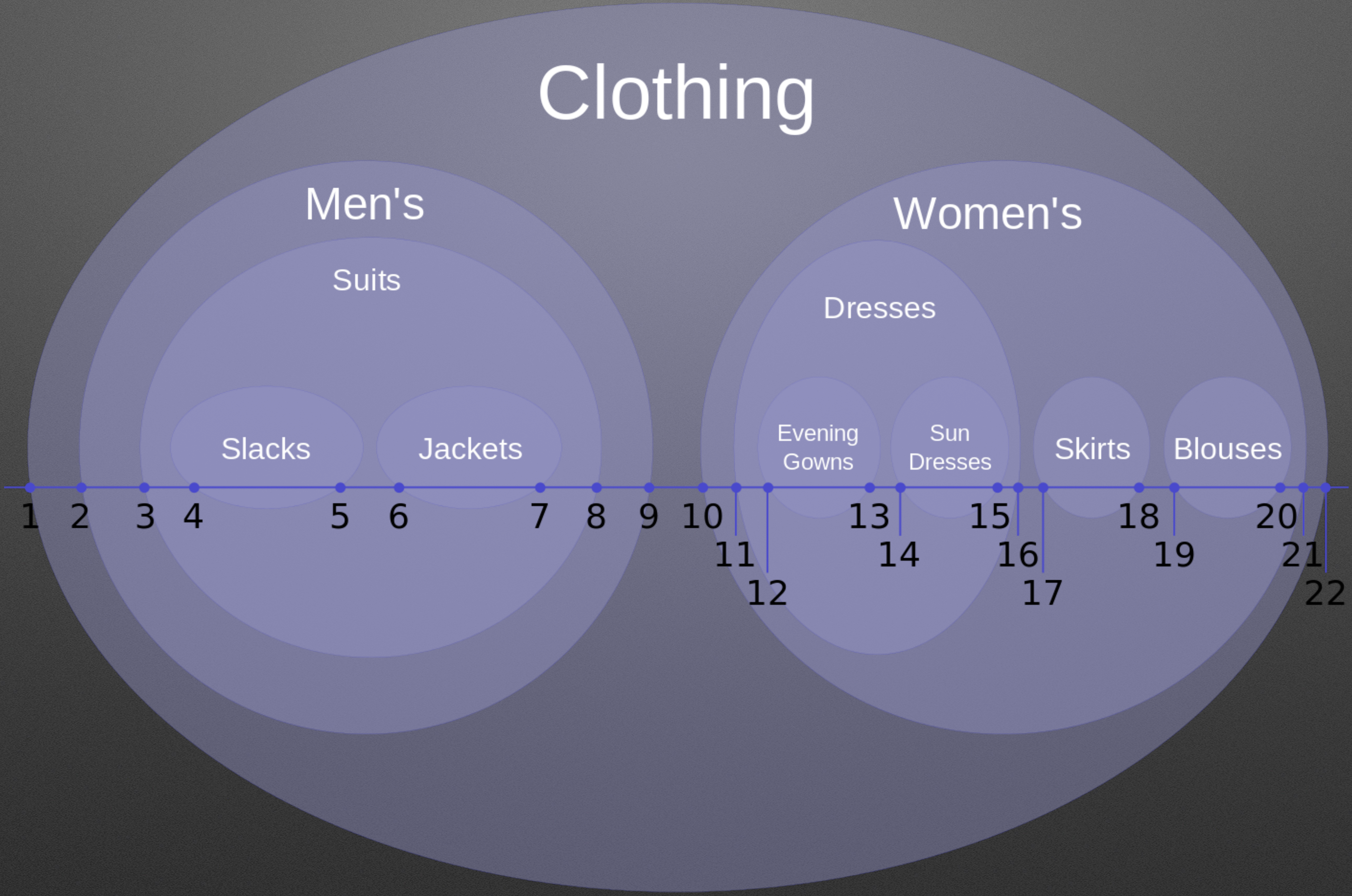
企业组织　　行政区域　　商品类目　　等级体系

文件分类　　资料归档　　授权体系　　……

A hierarchy

types of clothing

https://en.wikipedia.org/wiki/File:NestedSetModel.svg


Clothing
Men's
Suits
Slacks
Jackets
Women's
Dresses
Evening Gowns
Sun Dresses
Skirts
Blouses
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

2 主流方案

主流的存储方法

— 基于关系数据库（MySQL）

邻接列表模型 （The Adjacency List Model）

邻接列表模型

上述数据模型在关系数据库MySQL的表结构数据存储

通常如右图所示：

```
+----+-----------------+-----+
| id | node            | pid |
+----+-----------------+-----+
|  1 | Clothing        |   0 |
|  2 | Men's           |   1 |
|  3 | Women's         |   1 |
|  4 | Suits           |   2 |
|  5 | Dresses         |   3 |
|  6 | Skirts          |   3 |
|  7 | Blouses         |   3 |
|  8 | Slacks          |   4 |
|  9 | Jackets         |   4 |
| 10 | Evening Gowns   |   5 |
| 11 | Sun Dresses     |   5 |
+----+-----------------+-----+
```

缺陷

查找子树？查找祖谱？查找深度？完整数据？

3 改进方案

改进的前序遍历树模型

— 基于关系数据库（MySQL）

The Nested Set Model

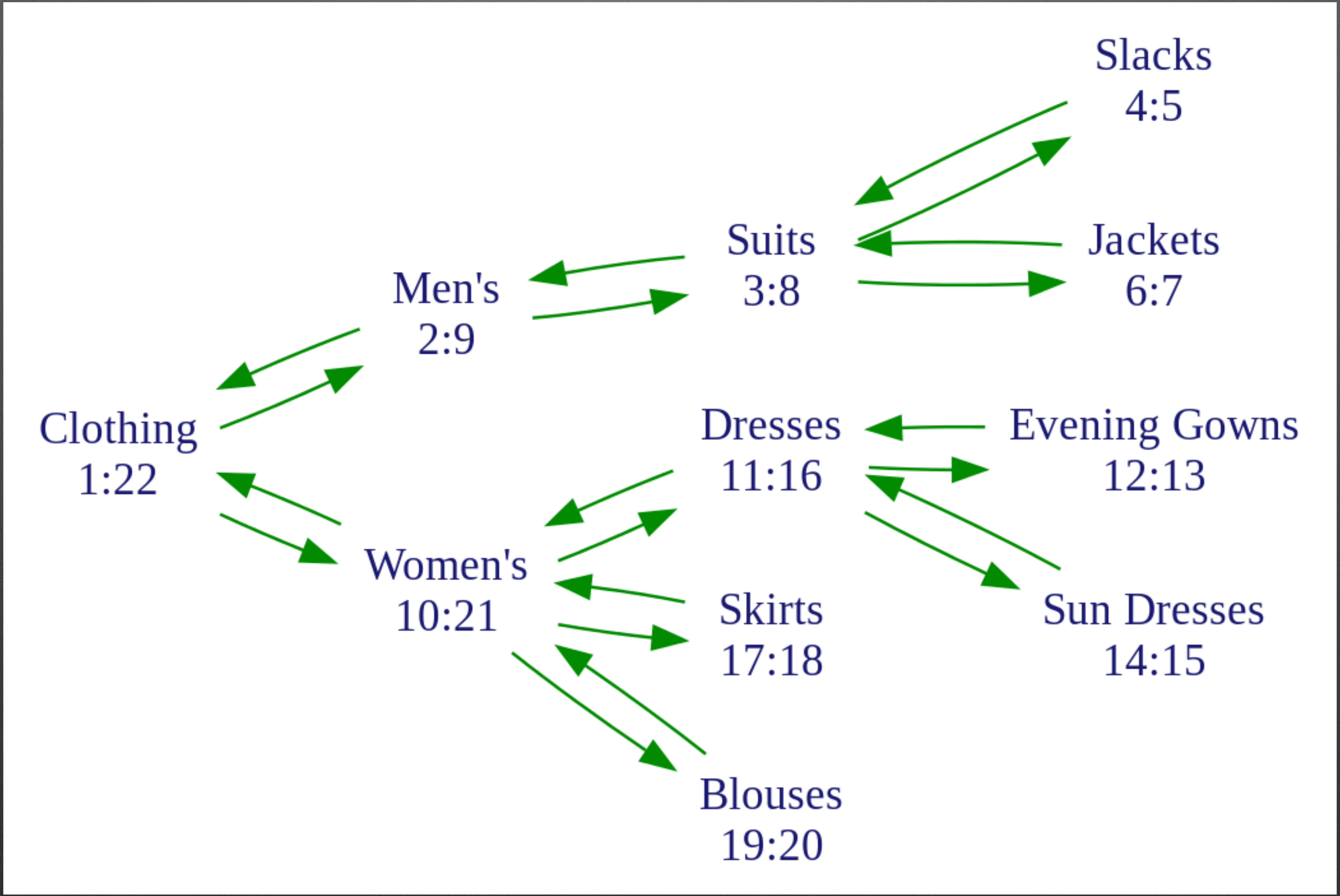
https://en.wikipedia.org/wiki/Nested_set_model

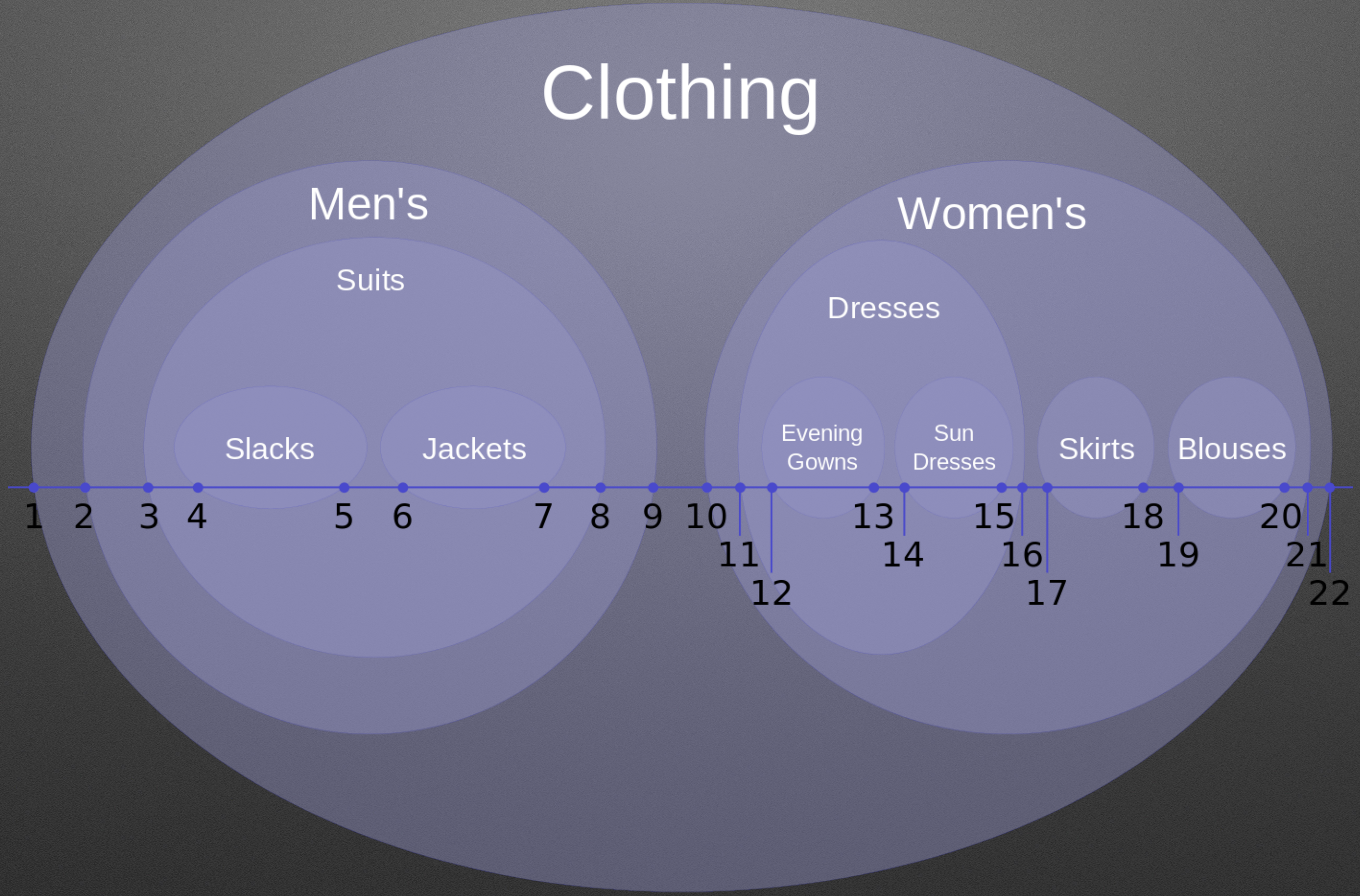
A hierarchy

types of clothing

https://en.wikipedia.org/wiki/File:Clothing-hierarchy-traversal-2.svg

The numbering assigned by tree traversal


Slacks 4:5
Suits 3:8
Jackets 6:7
Men's 2:9
Clothing 1:22
Dresses 11:16
Evening Gowns 12:13
Women's 10:21
Skirts 17:18
Sun Dresses 14:15
Blouses 19:20



Clothing
Men's
Suits
Slacks
Jackets
Women's
Dresses
Evening Gowns
Sun Dresses
Skirts
Blouses
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```
+----+---------------+-----+-------+-----+-----+
| id | node          | pid | depth | lft | rgt |
+----+---------------+-----+-------+-----+-----+
|  1 | Clothing      |   0 |     0 |   1 |  22 |
|  2 | Men's         |   1 |     1 |   2 |   9 |
|  3 | Women's       |   1 |     1 |  10 |  21 |
|  4 | Suits         |   2 |     2 |   3 |   8 |
|  5 | Dresses       |   3 |     2 |  11 |  16 |
|  6 | Skirts        |   3 |     2 |  17 |  18 |
|  7 | Blouses       |   3 |     2 |  19 |  20 |
|  8 | Slacks        |   4 |     3 |   4 |   5 |
|  9 | Jackets       |   4 |     3 |   6 |   7 |
| 10 | Evening Gowns |   5 |     3 |  12 |  13 |
| 11 | Sun Dresses   |   5 |     3 |  14 |  15 |
+----+---------------+-----+-------+-----+-----+
```

数据库表设计SQL

```sql
CREATE DATABASE IF NOT EXISTS `treeDB`;
CREATE TABLE IF NOT EXISTS `treeDB`.`nested`(
  `id` INT NOT NULL COMMENT '节点ID',
  `node` VARCHAR(64) CHARACTER SET 'utf8' NOT NULL COMMENT '节点名称',
  `pid` INT NOT NULL COMMENT '父节点ID',
  `depth` INT NOT NULL COMMENT '深度Level',
  `lft` INT NOT NULL COMMENT '左值',
  `rgt` INT NOT NULL COMMENT '右值',
  PRIMARY KEY (`id`),
  INDEX `depth_index` (`depth` ASC),
  INDEX `lft_index` (`lft` ASC),
  INDEX `rgt_index` (`rgt` ASC))
ENGINE = InnoDB DEFAULT CHARACTER SET = utf8 COMMENT = '前序遍历树模型表';
```

常用操作SQL

```
#插入根节点 (Adding New Nodes)
SET @vid:=1,@vnode:='Clothing',@vpid:=0,@vdepth:=0,@vlft:=1,@vrgt:=2;
INSERT INTO `treeDB`.`nested`(`id`,`node`,`pid`,`depth`,`lft`,`rgt`)
 VALUES(@vid,@vnode,@vpid,@vdepth,@vlft,@vrgt);
```

常用操作SQL

```
#插入节点 (Adding New Nodes)  —— First Child
SET @vid:=2,@vnode:='Men\'s',@vpid:=1;
SELECT @vprgt:=rgt,@vpdepth:=depth
 FROM `treeDB`.`nested` WHERE id=@vpid;
UPDATE `treeDB`.`nested`
 SET rgt=CASE WHEN rgt>=@vprgt THEN rgt+2 END,
     lft=CASE WHEN lft>=@vprgt THEN lft+2 END;
INSERT INTO `treeDB`.`nested`(`id`,`node`,`pid`,`depth`,`lft`,`rgt`)
 VALUES(@vid,@vnode,@vpid,@vpdepth+1,@vprgt,@vprgt+1);
```

常用操作SQL

```
#插入节点 (Adding New Nodes)  —— Not First Child
SET @vid:=3,@vnode:='Women\'s',@vlid:=1;
SELECT @vlpid:=pid,@vlrgt:=rgt,@vldepth:=depth
 FROM `treeDB`.`nested` WHERE id=@vlid;
UPDATE `treeDB`.`nested`
 SET rgt=CASE WHEN rgt>@vlrgt THEN rgt+2 END,
     lft=CASE WHEN lft>@vlrgt THEN lft+2 END;
INSERT INTO `treeDB`.`nested`(`id`,`node`,`pid`,`depth`,`lft`,`rgt`)
 VALUES(@vid,@vnode,@vlpid,@vldepth,@vlrgt+1,@vlrgt+2);
```

常用操作SQL

```
#返回某结点的子树 (Retrieving a Full/Subordinates Tree)
                                  – Given Parent Node Index
SET @vid:=2;
SELECT Child.id,Child.node,Child.pid,Child.depth,Child.lft,Child.rgt
 FROM `treeDB`.`nested` AS Parent,`treeDB`.`nested` AS Child
 WHERE Child.lft BETWEEN Parent.lft AND Parent.rgt
 AND Parent.id=@vid
 ORDER BY Child.id;
```

常用操作SQL

```
#返回某结点的祖谱路径 (Retrieving a Single Path)
                               — Given Child Node Index
SET @vid:=11;
SELECT Parent.id,Parent.node,Parent.pid,Parent.depth,Parent.lft,Parent.rgt
 FROM `treeDB`.`nested` AS Parent,`treeDB`.`nested` AS Child
 WHERE Child.lft BETWEEN Parent.lft AND Parent.rgt
 AND Child.id=@vid
 ORDER BY Child.id;
```

常用操作SQL

```
#返回所有的叶子节点 (Retrieving all the Leaf)
SELECT id,node,pid,depth,lft,rgt
 FROM `treeDB`.`nested` WHERE rgt=lft+1;
```

常用操作SQL

```
#删除节点 (Deleting Nodes)
SET @vid:=10;
SELECT @vlft:=lft,@vrgt:=rgt,@vwidth:=rgt-lft+1
 FROM `treeDB`.`nested` WHERE id=@vid;
DELETE FROM `treeDB`.`nested` WHERE lft BETWEEN @vlft AND @vrgt;
UPDATE `treeDB`.`nested`
 SET rgt=CASE WHEN rgt>@vrgt THEN rgt-@vwidth END,
     lft=CASE WHEN lft>@vrgt THEN lft-@vwidth END;
```

Thanks

To Be Continued...