

## Case study 12

---

# Complement on Kings of France

**Objective.** This document is a complement to case study *Kings of France - Part 2*. It details the development of graphical representation of the data of KINGS.db database with the vector graphics engine of SQLfast.

## 1. Graphical representation of king genealogy - version 1

Instead of merely printing member data as a pure text as we did in the previous section, it would be nicer to **draw** the tree as we may find it in some web sites.

We will use the SQLdraw engine to present the data in a graphical way. We first generate an SQLdraw script file representing the tree, then we render it with the `showDrawing` statement.

Let us begin with a first draft such as that of Figure 12.2. We can build it very simply by drawing, for each member, two points located at the same position:

- an *ornemental* point represented by a small black diamond, without label; a point model 31 is fine<sup>1</sup> (Figure 12.1),
- a point represented by a simple dash, followed by the name and title of the member; here, we choose a point model 68.

---

1. A point is defined by its coordinates, in horizontal and vertical pixels, by its type and by a label, possibly empty. The illustrated catalogue of SQLdraw points is presented in Chapter 21 of the Tutorial.

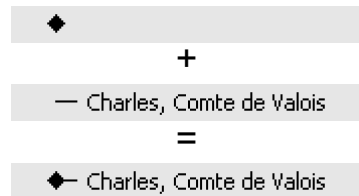


Figure 12.1 - Detail of the representation of a member

The first five members will be drawn by the following script.

```

area 370,780
points " ",1,"black","black",68
10,10,"Robert, Comte de Clermont"
points " ",1,"black","black",31
10,10,""

points " ",1,"black","black",68
20,25,"Louis I, Duc de Bourbon"
points " ",1,"black","black",31
20,25,""

points " ",1,"black","black",68
30,40,"Jacques I, Comte de La Marche"
points " ",1,"black","black",31
30,40,""

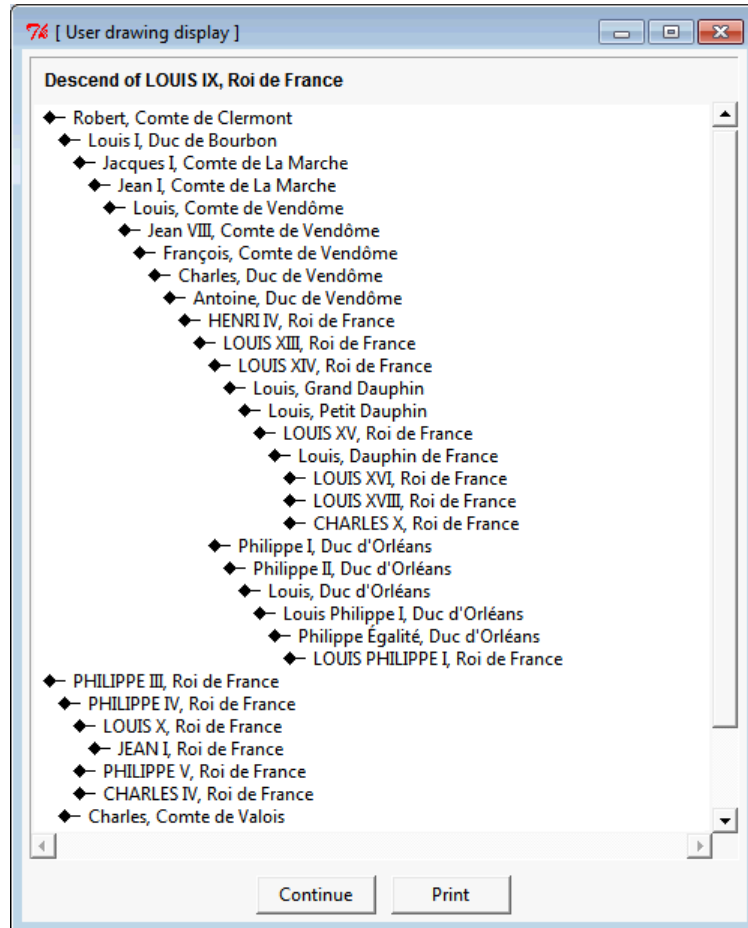
points " ",1,"black","black",68
40,55,"Jean I, Comte de La Marche"
points " ",1,"black","black",31
40,55,""

points " ",1,"black","black",68
50,70,"Louis, Comte de Vendôme"
points " ",1,"black","black",31
50,70,""

```

SQLdraw directive **area** specifies the dimensions of the drawing space. The (x,y) coordinates are measured in pixels and the origin of this space is (0,0) at the top-left corner. The first son of Louis IX, Robert, Comte de Clermont, is positioned at (10,10), the next members being drawn 10 point to the right of their father and 15 point downward from the previous one. Directive **points** starts a group of points (only one point here) by defining its thickness (1), its color (black) and its type (68 and 31). A group is assigned a label, here a simple space character " ", which can be ignored. An individual point is defined by its x, y coordinates and its label.

Script 12.1 builds and displays the descent of a selected member. Its principle derives from that of Script 12.2.



**Figure 12.2** - The descent of King LOUIS IX (excerpt) - First version

As it is used in several other queries, TREE is transformed into an independent view. For a much larger source MEMBER table, building a temporary table would probably be more efficient (*created once, used many times*).

The text of the SQLdraw script is generated into variable draw [3], then stored in file KINGdrawing.draw [6,7]. We choose to generate the script through an SQL query [5]. To avoid column names and tabular layout, we apply the elementary List output format [2]. This query, which produces four SQLdraw lines for each row, appears to be fairly complicated. To simplify its writing, we define a series of shorthands [4], through variables defining the size of the tree, in pixels (maxX, maxY), the expression of the position of each member in the tree, in pixels (coordX, coordY), and, for each member, the four lines defining the two points (header68, point68, header31, point31). Finally, the SQLdraw script is rendered graphically [8].

Printed 6/6/23

```

ask pid = member:[!select PiD from MEMBER order by PiD];
extract Nam,Tit = select Name,Title
                  from MEMBER where PiD = '$pid$';

create temp view TREE as
with recursive PATHS (Len,Father,PiD,Path) as
(select 0,'$pid$',PiD,'.'||'$pid$'||'.'||PiD||'. '
 from MEMBER where Father = '$pid$'
 union all
 select P.Len+1,P.PiD,M.PiD,P.Path||M.PiD||'. '
 from PATHS P,MEMBER M
 where P.PiD = M.Father
 )
select Len,Father,PiD,Path from PATHS;

```

```

execSQL UTIL-SELECT-parameters-for-Lists.sql; [2]
outputOpen draw.var; [3]
-- Some shorthands -- [4]
set maxX = max((Len+1)*10);
set maxY = max((select count(1) from TREE T2
                  where T2.Path < T.Path)*15)+10;
set coordX = cast((Len+1)*10 as char);
set coordY = cast((select count(1) from TREE T2
                  where T2.Path < T.Path)*15+10 as char);
set header68 = 'points " ",1,"black","black",68@n';
set point68 = '$coordX$|'|'$coordY$|'|'|M.Name|'|',
              '|M.Title|'|"@n';
set header31 = 'points " ",1,"black","black",31@n';
set point31 = '$coordX$|'|'$coordY$|'|','';
select 'area '|cast($maxX$+200 as char)|'|',
       '|cast($maxY$+50 as char) as Area from TREE T;
-- The generation query --
select $header68$|'$point68$|'$header31$|'$point31$ [5]
from TREE T,MEMBER M where T.PiD = M.PiD order by Path;
outputOpen KINGdrawing.draw; [6]
write $draw$; [7]
showDrawing KINGdrawing.draw = [/x450/y750Descent of $Nam$]; [8]

```

**Script 12.1** - Extracting the descent of a member (simple graphical view)

## 2. Graphical representation of king genealogy - version 2

Now, we will improve the drawing in three aspects. First, we add the *life period* of the members and, for kings, the *reign period*. Secondly, we color the king lines in blue (nobles and royal family members were said to have *blue blood*!). Thirdly, we draw connectors between members in order to make *father-son* relationships more explicit. To summarize, we propose to generate the drawing of Figure 12.4, a detail of which is shown in Figure 12.3.



**Figure 12.3** - Detail of the representation of members, their status and their father-son relationships

Though the procedure could be developed as an extension of the former procedure, we will implement it in a slightly different way. First, view TREE will be transformed into a real (temporary) table. This table already includes all the coordinates of the graphic points, to avoid further computation at generation time. Secondly, the central query will not rely on variables but will be given a layout that evokes the list of eight lines to generate for each member. In addition, the first line must be prepared in a special way, since it has no parent line.

Script 12.2 creates table TREE, with five new columns: **Seq**, a sequence number generated automatically, **X** and **Y** the current coordinates and **FX** and **FY** the coordinates of the father. Note that the *order by* clause is necessary to let the values of **Seq** follow the *depth first* traversal of the tree in construction. Column **X** is computed at load time as  $(Len + 1) * 10$ . Column **Y**, **FX** and **FY** are set through an update query. Column **Y** is derived from the value of **Seq**, while the values of columns **FX** and **FY** are extracted from the row of the father.

The main query is shown in Script 12.3. The sequence of elements that build each of the eight SQLdraw lines are identified by number [1] to [8]. The big *case-end* block noted [\*] controls the generation of the *first* and the *following* lines.

**Reign period and blue color:** in sequences [1], [2] and [3], they are generated when the title of the member is 'Roi de France'. The choice is expressed by three *case-end* function.

**Graphical links.** The link between a father and one of his son is drawn as a *polyline*, that is, sequence of connected segments. The SQLdraw directive **polyline** defines a label (ignored here), the thickness of the segments (here 1), the color of the segments (**black**) and the fill color (empty) if the polyline also is a polygon, which is meaningless here. The polyline comprises a vertical segments and a horizontal segment, defined by three points. To draw them, we must know the coordinates of the current member (columns **X**, **Y**) and those of his father (columns **FX**, **FY**).



```

30,115,"Charles, Comte de Valois (1270-1325)"
points " ",1,"black","black",31
30,115,""
polyline " ",1,"black",""
20,25
20,115
30,115

```

```

create temporary table TREE (
    Seq integer primary key autoincrement,
    Len integer, Father char(6),
    PiD char(6), Path varchar(1024),
    X integer, Y integer, FX integer, FY integer);
with recursive PATHS(Len,Father,PiD,Path) as ($Closure$)
insert into TREE (Len,Father,PiD,Path,X)
select Len+1,Father,PiD,Path,(Len+1)*10
from PATHS order by Path;
update TREE
set Y = Seq*15 - 5,
    FX = (select X from TREE T where PiD = TREE.Father),
    FY = (select Seq*15-5 from TREE T where PiD = TREE.Father);
extract maxX,maxY = select max(X)+320,max(Y)+40 from TREE;

```

**Script 12.2** - Descent of a member: preparing table TREE

## A procedural approach

The main query in Script 12.3, which packs in one statement all the logic of text generation may appear large and complex (it is, but basically because the text itself is complex), and therefore prone to reading and writing errors. It could be decomposed into a series of select queries, each addressing a specific pattern: first line, following lines, kings, other members.

Some readers may find a more procedural expression better suited. Script 12.4 is such a proposal. It replaces the large query of Script 12.3.

The procedure is structured as a large loop, traversing the TREE table.

The case *first/non-first* member is described through variable **first** (value 1 for *first* and 0 for *non-first*). This variable is initialized to 1, then is set to 0 at the end of the first (actually of all) iteration of the loop.

Whether a member is a king or not is described by variable **king** (value 1 for *king* and 0 for *non-king*), set by the SQL query of the loop header.

```

write area $maxX$, $maxY$;
select case when M.Title = 'Roi de France' [1]
      then 'points " ",1,"black","blue",68@n'
      else 'points " ",1,"black","black",68@n'
      end
  || cast(X as char) || ',' || cast(Y as char) || ',' [2]
  || M.Name || ',' || M.Title
  || ' (' || M.DateBirth
  || '-' || M.DateDeath || ')'
  || case when M.Title = 'Roi de France'
      then ' (' || M.DateReignBegin
      || '-'
      || M.DateReignEnd || ') "@n'
      else '"@n'
      end
  || case when M.Title = 'Roi de France' [3]
      then 'points " ",1,"blue","black",31@n'
      else 'points " ",1,"black","black",31@n'
      end
  || cast(X as char) || ',' || cast(Y as char) || ',' [4]
  || case when T.FX is null then ' ' [5]
      else '@n'
      || 'polyline " ",1,"black",""@n' [6]
      || cast(FX as char) || ',' [7]
      || cast(FY as char) || ',' "@n' [8]
      || cast(FX as char) || ',' [9]
      || cast(Y as char) || ',' "@n' [10]
      || cast(X as char) || ',' [11]
      || cast(Y as char) || ',' [12]
  end
from TREE T, MEMBER M where T.PiD = M.PiD order by Path;

```

**Script 12.3** - Descent of a member: generating the script of the full graphical view.  
[KING-20-Members-Full-Drawing.sql]

Both variants provide the same result, but a slight difference can be observed, notably for the longest descent, that of Hughes Capet. The procedural script takes more time to build the SQLdraw script: **1.125** seconds for the SQL loop in the procedural script against **0.031** seconds for the pure SQL query. Such a difference is not significant for small problems but may be an issue for larger ones. However, these figures depends on technical options of the SQLfast interpreter, options that may evolve in the future.

### Note

Script **King-Full-Drawing-with-Root.sql** is a minor extension of Script **King-Full-Drawing.sql** that also displays the root of the tree and the PiD value of each member. Considering that the *House* is represented by the first character of PiD,



the picture produced (available in file **Capetian-dynasty.jpg**) can be seen as the graphical equivalent of the source data.

```

for Seq,Len,Father,PiD,Path,X,Y,FX,FY,
  nam,tit,datB,datD,dateRB,dateRE,king
  = [select Seq,Len,M.Father,M.PiD,Path,X,Y,FX,FY,
      Name,Title,DateBirth,DateDeath,
      DateReignBegin,DateReignEnd,
      case when Title = 'Roi de France' then 1 else 0 end
    from TREE T, MEMBER M
    where T.PiD = M.PiD order by Path];
set color = black;
if ($king$ = 1) set color = blue;
write points " ",1,"black","$color$",68;
set line = $X$,$Y$,"$nam$", $tit$ ($datB$-$datD$);
if ($king$ = 1) set line = $line$ ($dateRB$-$dateRE$);
write $line$";
write points " ",1,"$color$","black",31;
write $X$,$Y$,"";
if ($first$ = 0);
  write polyline " ",1,"black","";
  write $FX$,$FY$,"";
  write $FX$,$Y$,"";
  write $X$,$Y$,"";
endif;
set first = 0;
endfor;

```

**Script 12.4** - Descent of a member: generating the script of the full graphical view through an SQLfast loop-based script

