TD1 : requêtes de chemin en JSON Path - correction

UCBL - département informatique - MIF24 BD NoSQL - 2024/2025

Objectif du TD : se familiariser avec les concepts des langages de chemins en pratiquant les requêtes JSON Path

Exercice 1 Interprétation SQL / JSON Path

On considère le document JSON de la figure 1. Pour chacune des expressions JSON Path suivantes, exprimer sa signification en français, puis l'évaluer pas à pas. On indiquera à chaque étape l'ensemble des valeurs obtenues. Si ces valeurs sont des nœuds internes (liste ou dictionnaire), on indiquera le numéro de la ligne dans le document correspondant au début de ce nœud.

1. `$[0].nom`
2. `$[1].albums[0].titre`
3. `$[*].albums[*].annee`
4. `$[*].albums[*] ?(@.pistes[*].titre == "Back in Black").annee`
5. `$[*].albums[*] ?(@.titre == @.pistes[*].titre).titre`

Exercice 2 Écriture de requêtes JSON Path

Donner, pour chacune des questions suivantes, une expression JSONPath qui permet d'y répondre. On supposera que le document interrogé est similaire au document de la figure 1.

1. Quel est le nom de chaque artiste/groupe ?
2. Quelle est la durée de chaque piste ?
3. Quel est le titre de la première piste de l'album "Back in Black" ?
4. Quels sont les titres des albums contenant une piste plus longue que `"06:00"` ?
5. Quelles sont les pistes plus longues que "Hells Bells" ?

Exercice 3 Interpréteur JSON Path

On souhaite implémenter un interpréteur (d'une sous-partie) de JSON Path. On rappelle dans les figures 2 et 3 la sémantique du langage, pour la partie expressions de chemin.

1. Proposer une structure JSON permettant de représenter une expression JSON Path
2. Coder la fonction `eval` (par exemple en Javascript ou en Python). On rappelle que les deux premiers arguments de la fonction sont des (sous-)documents JSON et que le dernier argument est l'expression à évaluer (codée dans la structure précédemment définie). On ne traitera pas les conditions `?( ... )` dans cette question.
3. Coder une fonction `evalCond` qui prend des arguments `(r,c,exprB)` et qui renvoie vrai si $(r, c) \models exprB$
4. Terminer le codage de `eval`

```
[
  {
    "nom": "AC/DC",
    "albums": [
      {
        "titre": "Highway to Hell",
        "annee": 1979,
        "pistes": [
          { "titre": "Highway to Hell", "duree": "03:26" },
          { "titre": "Girls Got Rhythm", "duree": "03:23" },
          { "titre": "Walk All Over You", "duree": "05:08" },
          { "titre": "Touch Too Much", "duree": "04:24" },
          { "titre": "Beating Around the Bush", "duree": "03:55" },
          { "titre": "Shot Down in Flames", "duree": "03:21" },
          { "titre": "Get It Hot", "duree": "02:34" },
          { "titre": "If You Want Blood (You ve Got It)", "duree": "04:32" },
          { "titre": "Love Hungry Man", "duree": "04:14" },
          { "titre": "Night Prowler", "duree": "06:13" }
        ]
      },
      {
        "titre": "Back in Black",
        "annee": 1980,
        "pistes": [
          { "titre": "Hells Bells", "duree": "05:12" },
          { "titre": "Shoot to Thrill", "duree": "05:17" },
          { "titre": "What Do You Do for Money Honey", "duree": "03:37" },
          { "titre": "Givin the Dog a Bone", "duree": "03:33" },
          { "titre": "Let Me Put My Love into You", "duree": "04:16" },
          { "titre": "Back in Black", "duree": "04:16" },
          { "titre": "You Shook Me All Night Long", "duree": "03:32" },
          { "titre": "Have a Drink on Me", "duree": "03:58" },
          { "titre": "Shake a Leg", "duree": "04:06" },
          { "titre": "Rock and Roll Ain t Noise Pollution", "duree": "04:16" }
        ]
      }
    ]
  },
  {
    "nom": "Mike Oldfield",
    "albums": [
      {
        "titre": "Tubular Bells",
        "annee": 1973,
        "pistes": [
          { "titre": "Tubular Bells, Part 1", "duree": "25:36" },
          { "titre": "Tubular Bells, Part 2", "duree": "23:20" }
        ]
      }
    ]
  }
]
```

FIG. 1 : Document JSON sur une collection d'albums musicaux

$$
\begin{aligned}
&eval(r, c, \texttt{\$}) = \{r\}\\
&eval(r, c, \texttt{@}) = \{c\}\\
&eval(r, c, exprC.champ) = \bigcup_{j \in eval(r,c,exprC)} acces(j, champ)\\
&eval(r, c, exprC.*) = \bigcup_{j \in eval(r,c,exprC)} valeurs(j)\\
&eval(r, c, exprC[n]) = eval(r, c, exprC.n)\\
&eval(r, c, exprC[*]) = eval(r, c, exprC.*)\\
&eval(r, c, exprC..*) = \bigcup_{j \in eval(r,c,exprC)} sousDoc(j)\\
&eval(r, c, exprC\texttt{?(}exprB\texttt{)}) = \{j \in eval(r, c, exprC) \mid (r, j) \models exprB\}
\end{aligned}
$$

FIG. 2 : Interprétation des expressions de chemin JSON Path

$$
\begin{aligned}
&(r, c) \models \texttt{true}\\
&(r, c) \models \texttt{!}exprB \text{ si } (r, c) \not\models exprB\\
&(r, c) \models exprB_1 \texttt{ \&\& } exprB_2 \text{ si } (r, c) \models exprB_1 \text{ et } (r, c) \models exprB_2\\
&(r, c) \models exprB_1 \texttt{ || } exprB_2 \text{ si } (r, c) \models exprB_1 \text{ ou } (r, c) \models exprB_2\\
&(r, c) \models exprC\ op\ val \text{ si } \exists j \in eval(r, c, exprC)\\
&\qquad\qquad\qquad\qquad \text{tel que } j\ op\ val \text{ est vrai}\\
&(r, c) \models exprC_1\ op\ exprC_2 \text{ si } \exists j_1 \in eval(r, c, exprC_1),\\
&\qquad\qquad\qquad\qquad\quad \exists j_2 \in eval(r, c, exprC_2)\\
&\qquad\qquad\qquad\qquad\quad \text{tels que } j_1\ op\ j_2 \text{ est vrai}\\
&(r, c) \models \texttt{exists(}exprC\texttt{)} \text{ si } eval(r, c, exprC) \neq \emptyset
\end{aligned}
$$

FIG. 3 : Satisfaction des conditions JSON Path

Réponses de l'Exercice 1

1. Le nom du premier artiste/groupe.

   `$` $\rightsquigarrow$ 1
   `$[0]` $\rightsquigarrow$ 2
   `$[0].nom` $\rightsquigarrow$ `"AC/DC"`

   Ligne de commande : `jq ".[0].nom" ex1-1.json`

2. Le titre du premier album du deuxième artiste/groupe.

   `$` $\rightsquigarrow$ 1
   `$[1]` $\rightsquigarrow$ 39
   `$[1].albums` $\rightsquigarrow$ 41
   `$[1].albums[0]` $\rightsquigarrow$ 42
   `$[1].albums[0].titre` $\rightsquigarrow$ `"Tubular Bells"`

   Ligne de commande : `jq ".[1].albums[0].titre" ex1-1.json`

3. L'année de chaque album.

   `$` $\rightsquigarrow$ 1
   `$[*]` $\rightsquigarrow$ 2, 39
   `$[*].albums` $\rightsquigarrow$ 4, 41
   `$[*].albums[*]` $\rightsquigarrow$ 5, 21, 42
   `$[*].albums[*].annee` $\rightsquigarrow$ 1979, 1980, 1973

   Ligne de commande : `jq ".[].albums[].annee" ex1-1.json`

4. L'année du ou des albums contenant une piste intitulée "Back in Black".

   `$` $\rightsquigarrow$ 1
   `$[*]` $\rightsquigarrow$ 2, 39
   `$[*].albums` $\rightsquigarrow$ 4, 41
   `$[*].albums[*]` $\rightsquigarrow$ 5, 21, 42
   `$[*].albums[*] ?(@.pistes[*].titre == "Back in Black")`
   `@`= 5 :
   `@.pistes` $\rightsquigarrow$ 8
   `@.pistes[*]` $\rightsquigarrow$ 9, 10, . . . , 17, 18
   `@.pistes[*].titre` $\rightsquigarrow$ `"Highway to Hell","Girls Got Rhythm",...,`
   `"Love Hungry Man","Night Prowler"`
   `@.pistes[*].titre == "Back in Black"` $\rightsquigarrow$ `false, false, ..., false, false`
   5 n'est **pas** conservé
   `@`= 21 :
   `@.pistes` $\rightsquigarrow$ 24
   `@.pistes[*]` $\rightsquigarrow$ 25, . . . , 30, . . . , 34
   `@.pistes[*].titre` $\rightsquigarrow$ `"Hells Bells",...,"Back in Black",...,`
   `"Rock and Roll Ain t Noise Pollution"`
   `@.pistes[*].titre == "Back in Black"` $\rightsquigarrow$ `false, ..., true, ..., false`
   21 est conservé
   `@`= 42 :
   `@.pistes` $\rightsquigarrow$ 45
   `@.pistes[*]` $\rightsquigarrow$ 46, 47
   `@.pistes[*].titre` $\rightsquigarrow$ `"Tubular Bells, Part 1","Tubular Bells, Part 2"`
   `@.pistes[*].titre == "Back in Black"` $\rightsquigarrow$ `false, false`
   42 n'est **pas** conservé
   $\rightsquigarrow$ 21
   `$[*].albums[*] ?(@.pistes[*].titre == "Back in Black") .annee` $\rightsquigarrow$ 1980

Ligne de commande :
`jq '.[].albums[] | select(.pistes[].titre=="Back in Black").annee' ex1-1.json`

5. Le titre des albums dont une piste est intitulée comme cet album.
   `$` $\rightsquigarrow 1$
   `$[*]` $\rightsquigarrow 2, 39$
   `$[*].albums` $\rightsquigarrow 4, 41$
   `$[*].albums[*]` $\rightsquigarrow 5, 21, 42$
   `$[*].albums[*] ?(@.titre == @.pistes[*].titre)`
   `@`$= 5$:
   `@.titre`$\rightsquigarrow$`"Highway to Hell"`
   `@.pistes[*].titre`$\rightsquigarrow$`"Highway to Hell", "Girls Got Rhythm",...,`
   `"Love Hungry Man","Night Prowler"`
   5 est conservé car au moins un élément donné par `@.titre` correspond
   (i.e. est `==`) à un élément donné par `@.pistes[*].titre`
   `@`$= 21$:
   `@.titre`$\rightsquigarrow$`"Back in Black"`
   `@.pistes[*].titre`$\rightsquigarrow$`"Hells Bells",...,"Back in Black",...,`
   `"Rock and Roll Ain t Noise Pollution"`
   21 est conservé car au moins un élément donné par `@.titre` correspond
   (i.e. est `==`) à un élément donné par `@.pistes[*].titre`
   `@`$= 42$:
   `@.titre`$\rightsquigarrow$`"Tubular Bells"`
   `@.pistes[*].titre`$\rightsquigarrow$`"Tubular Bells, Part 1","Tubular Bells, Part 2"`
   42 n'est **pas** conservé, car aucun un élément donné par `@.titre` ne correspond
   (i.e. est `==`) à un élément donné par `@.pistes[*].titre`
   $\rightsquigarrow 5, 21$
   `$[*].albums[*] ?(@.titre == @.pistes[*].titre) .titre`
   $\rightsquigarrow$ `"Highway to Hell", "Back in Black"`

   Ligne de commande :
   `jq ".[].albums[] | select(.pistes[].titre==.titre).titre" ex1-1.json`

Réponses de l'Exercice 2

Les réponses sont données SQL / JSONPath.

1. `$[*].nom`
   Ligne de commande : `jq ".[].nom" ex1-1.json`

2. `$[*].albums[*].pistes[*].duree` ou `$.**.duree`
   Ligne de commande : `jq ".[].albums[].pistes[]" ex1-1.json`

3. `$[*].albums[*] ?(@.titre == "Back in Black") .pistes[0].titre`
   Ligne de commande : `jq ".[].albums[]|select(.titre==\"Back in Black\") .pistes[0].titre" ex1-1.json`

4. `$[*].albums[*] ?(@.pistes[*].duree > "06:00") .titre`
   Ligne de commande : `jq '.[].albums[]|select(.pistes[].duree>"06:00").titre' ex1-1.json`
   (attention, doublons)

5. `$[*].albums[*].pistes[*] ?(@.duree > $.** ?(@.titre == "Hells Bells").duree )`
   Ligne de commande : `jq` ne semble pas posséder la construction `$`. Il possède cependant un mécanisme plus puissant permettant d'itérer sur le résultat en déclarant une variable[1]. On peut utiliser ce mécanisme

[1] *c.f.* *https://stedolan.github.io/jq/manual/#Advancedfeatures*

Expression	Structure
`$`	`{ "op": "$" }`
`@`	`{ "op": "@" }`
exprC.*champ*	`{ "op": ".", "expr":` *exprC*`, "field": "`*champ*`" }`
exprC[*champ*]	*idem exprC*.*champ*
exprC.∗	`{ "op": "*", "expr":` *exprC* `}`
exprC[∗]	*idem exprC*.∗
exprC..∗	`{ "op": "..", "expr":` *exprC* `}`
exprC`?(`*exprB*`)`	`{ "op": "?", "expr":` *exprC*`, "cond":` *exprB* `}`

TAB. 1 : Structure pour les expressions JSONPath

Condition	Structure
`true`	`{ "op": "true" }`
`false`	`{ "op": "false" }`
`!` *exprB*	`{ "op": "!", "expr":` *exprB* `}`
exprB$_1$ `&&` *exprB*$_2$	`{ "op": "&&", "expr1":` *exprB*$_1$`, "expr2":` *exprB*$_2$ `}`
exprB$_1$ `||` *exprB*$_2$	`{ "op": "||", "expr1":` *exprB*$_1$`, "expr2":` *exprB*$_2$ `}`
exprC op val	`{ "op": "cmp_val", "cmp":` *op*`, "val":` *val*`, "expr":` *exprC* `}`
exprC$_1$ *op exprC*$_2$	`{ "op": "cmp_expr", "cmp":` *op*`, "expr1":` *exprC*$_1$`, "expr2":` *exprC*$_2$ `}`
`exists(`*exprC*`)`	`{ "op": "exists", "expr":` *exprC* `}`

TAB. 2 : Structure pour les conditions JSONPath

pour mettre la racine du document dans une variable, ce qui donne la command suivante (à mettre sur une seule ligne) :

```
jq '. as $root | .[].albums[].pistes[]
                | select(.duree > ($root[].albums[].pistes[]
                                   | select(.titre == "Hells Bells")
                                   | .duree ) )
                | .titre'
   ex1-1.json
```

Réponses de l'Exercice 3

1. On peut par exemple prendre une structure comme celle présentée dans les tables 1 et 2