Le selezioni in cascata

La funzionalità di selezione in cascata[1] consiste nella possibilità di esprimere, con una sola espressione di ricerca, più selezioni da compiere in sequenza potendo riflettere sulle successive l'esito delle precedenti. Ciascuna di esse produce infatti un file di selezione, quindi un semilavorato della selezione complessiva che intendiamo compiere, che possiamo riutilizzare nelle selezioni successive.

Perché si usa

Lo scopo della selezione in cascata è quella di semplificare il compito del server quando le operazioni sono complesse e ridurre l'*overhead* dato dalla comunicazione *Client* ↔ *Server*.

Immaginiamo ad esempio di avere due unità informative distinte, *extension* e *container* in cui l'unità *container* abbia un riferimento ad un identificatore delle unità *extension*. Volendo selezionare tutti i documenti di *container* che hanno una certa descrizione o che fanno riferimento ad una certa *extension* avente un certo titolo dovremo:

- Selezionare tutte le unità informative *extension* aventi il titolo richiesto
- Selezionare tutte le unità informative *container* aventi la descrizione desiderata ovvero aventi un riferimento ad uno dei documenti di cui alla selezione precedente.

In condizioni normali compiremmo la prima selezione di seguito esemplificata

`[xml, /extension/@title]="test"`

Questa selezione torna produce un semilavorato che è, ad esempio, `3se123456789.tmp`. In seguito, utilizzeremo questo sottoprodotto per perfezionare la selezione successiva come nell'esempio seguente.

```
[xml,/container/@description]="test" OR
[xml,/container/extension_ref/@id]={xml,/extension/@id;3se123456789.tmp}
```

Questo comporta l'esecuzione di due diverse operazioni da parte dell'applicazione. Lo stesso può ora essere eseguito con un operazione sola.

Come su usa

Abbiamo appena identificato le due diverse selezioni. Quello che ora ci serve sapere è che possiamo usare dei modificatori speciali della frase di ricerca che sono di seguito elencati:

Modificatore	Descrizione
<?THEN?>	E' il modificatore che consente di frazionare in parti distinte una frase di ricerca complessa. Le singole parti verranno prese in esame da sinistra verso destra ed eseguite in tale ordine.
<?SEL$_{numero}$?>	Indica una selezione precedentemente eseguita. Va da se che non può essere utilizzata nella prima porzione della selezione[2] ne si possono usare senza il dovuto ordine numerico. L'espressione è in base '0' quindi: <?SEL0?> rappresenta il semilavorato della prima selezione <?SEL1?> rappresenta il semilavorato della seconda selezione ...e così via sino a <?SEL9?> che è il massimo valore ammesso. In sostanza, quindi, nell'ennesima porzione della selezione non potrò utilizzare le i modificatori che indicano semilavorati non appartenenti alle `n-1` selezioni precedenti.

(Il case con il quale vengono espressi questi modificatori non è importante, il server provvede a riconoscerli indipendentemente da esso)

La precedente espressione verrà quindi tradotta come segue:

```
[xml, /extension/@title]="test" <?THEN?> [xml,/container/@description]="test" OR
[xml,/container/extension_ref/@id]={xml,/extension/@id;<?SEL0?>}
```

In questo modo, il semilavorato della prima selezione viene automaticamente utilizzato per la risoluzione della seconda.

Il risultato finale delle operazioni svolte sarà l'esito della selezione di estrema destra. I semilavorati prodotti sino a quel momento non verranno messi a disposizione del *Cilent*

Esempi e Dettagli

I dettagli della sintassi da usare possono essere più evidenti con alcuni esempi:

1

La frase di selezione...

```
[campo]=valore1 and valore2 <?THEN?> [?SEL]="<?SEL0?>" AND [campo]=valore2 or valore3
```

... può essere espressa anche come...

```
[campo]=valore1 and valore2 <?THEN?> [campo]=valore2 or valore3 <?THEN?> [?SEL]="<?SEL0?>" AND
[?SEL]="<?SEL1?>"
```

...nel primo caso il semilavorato della prima selezione viene posto in AND con la seconda compiendone il raffinamento mentre nel secondo caso le due selezioni vengono compiute separatamente e poi combinate in AND portando, in sostanza allo stesso risultato.

Esempio 2

Analogamente la selezione espressa di seguito...

```
[campo]=valore1 and valore2 <?THEN?> [?SEL]="<?SEL1?>" AND [campo]=valore2 or valore3
```

...è errata in quanto nella seconda parte della selezione si possono usare solo i semilavorati delle selezioni già compiute[3].

Esempio 3

In fine si noti come le espressioni...

```
"<?SEL1?>"
```

... e ...

```
<?SEL1?>
```

...siano di fatto corrispondenti in quanto il server provvede a *circondare* con i doppi apici l'identificatore della selezione che rappresenta il semilavorato.

Considerazioni sulle performance

Introduzione

Risulta evidente sin dalla prima lettura, che questo sistema di selezione si applichi solo in casi piuttosto particolari ma no va esclusa la possibilità di farne uso per incrementare le performance di selezioni complesse.
Vale la pena di sottolineare alcuni aspetti:

- Le performance in selezione sono tanto migliore quando le ricerche prevedono solo ed esclusivamente operatori AND[4] oppure solo ed esclusivamente operatori OR. In questi due casi la selezione è quanto più possibile rapida.
- Se la selezione non può essere espressa nel modo suddetto, la ricerca subisce un degrado prestazionale che è strettamente dipendente dal numero di chiavi coinvolte. Questo è vero sino a quando la ricerca non comprende operatori di negazione[5] o disuguaglianza[6].
- Quando la ricerca espressa presenta operatori di negazione o disuguaglianza si ricade nel peggiore dei casi, ovvero nella selezione più gravosa che il server possa compiere.

Detto questo risulta evidente come la selezione...

```
[campo]=valore1 and valore2 or valore3
```

...ricade nel secondo dei casi. Ricorrendo alla forma in cascata, potremmo avere...

```
[campo]=valore1 and valore2 <?THEN?> [?SEL]="<?SEL0?>" OR [campo]=valore3
```

...che cambia completamente il proprio aspetto in quanto la prima porzione è una ricerca che prevede esclusivamente operatori AND mentr la seconda porzione è espressa esclusivamente con operatori OR.
Questo non vuol dire che questa trasformazione nella selezione debba rappresentare una miglioria nelle performance. Una delle selezioni intermedie di un'espressione così articolata, ad esempio, potrebbe produrre un semilavorato molto ampio e quindi avere una perdita di performance nella stesura del file temporaneo corrispondente.

I casi più complessi

Proviamo ora a vedere un caso lievemente differente, l'opposto del precedente, con la selezione...

```
[campo]=(valore1 or valore2) and valore3
```

...in questo caso, per poter frazionare la selezione secondo il criterio espresso avremmo...

```
[campo]=(valore1 or valore2) <?THEN?> [?SEL]="<?SEL0?>" AND [campo]=valore3
```

...che apparentemente è identico al caso precedente ma ciò non è vero. In questo caso viene prima compiuta una ricerca in OR che potrebbe utilizzare catene di riferimenti molto lunghe e giungere a selezionare un elevato numero di documenti. Questo semilavorato, poi, verrebbe posto in AND con un ulteriore valore. Se quest'ultimo avesse una catena di riferimenti molto breve avremmo una selezione che si conclude molto rapidamente ma solo dopo aver speso molto tempo con la prima.
In questo scenario, ovvero ammettendo che il termine `valore3` sia molto selettivo, anche se la cosa può apparire un po' poco intuitiva, la selezione potrebbe avere la seguente forma...

```
[campo]=(valore1 and valore3) <?THEN?> [campo]=(valore2 and valore3) <?THEN?> [?SEL]="<?SEL0?>" OR
[?SEL]="<?SEL1?>"
```

In questo modo la prima e la seconda selezione verrebbero eseguite molto rapidamente[7] e la terza, questa volta tutta in OR, dovrebbe miscelare due sottoprodotti molto brevi.

Conclusioni

L'introduzione di questo nuovo sistema di selezione può risultare particolarmente utile per:

1. Esprimere condizioni in cascata che non sarebbero altrimenti esprimibili se non con una maggiore necessità di elaborazione sul fronte applicativo.
2. Esprimere selezioni che si potrebbero esprimere in modo *standard* per avvalersi di prestazioni migliorate.

Se nel primo caso questa forma di selezione è senza dubbio la migliore, la più indicata, nel secondo è opportuno che lo sviluppatore

abbia piena cognizione di ciò che sta facendo ed una conoscenza approfondita dell'archivio che gli consenta di valutare l'effettiva opportunità di adottare un simile stile.

1)
Disponibile a partire dal server eXtraWay versione 23.0.0.0 e successive

2)
Non essendo presente alcun semilavorato pregresso

3)
Nel nostro caso solo <?SEL0?>

4)
Ovvero di adiacenza

5)
NOT e NON

6)
<>

7)
In virtù della brevità della catena dei riferimenti del termine `valore3` e dell'uso dell'opertore AND