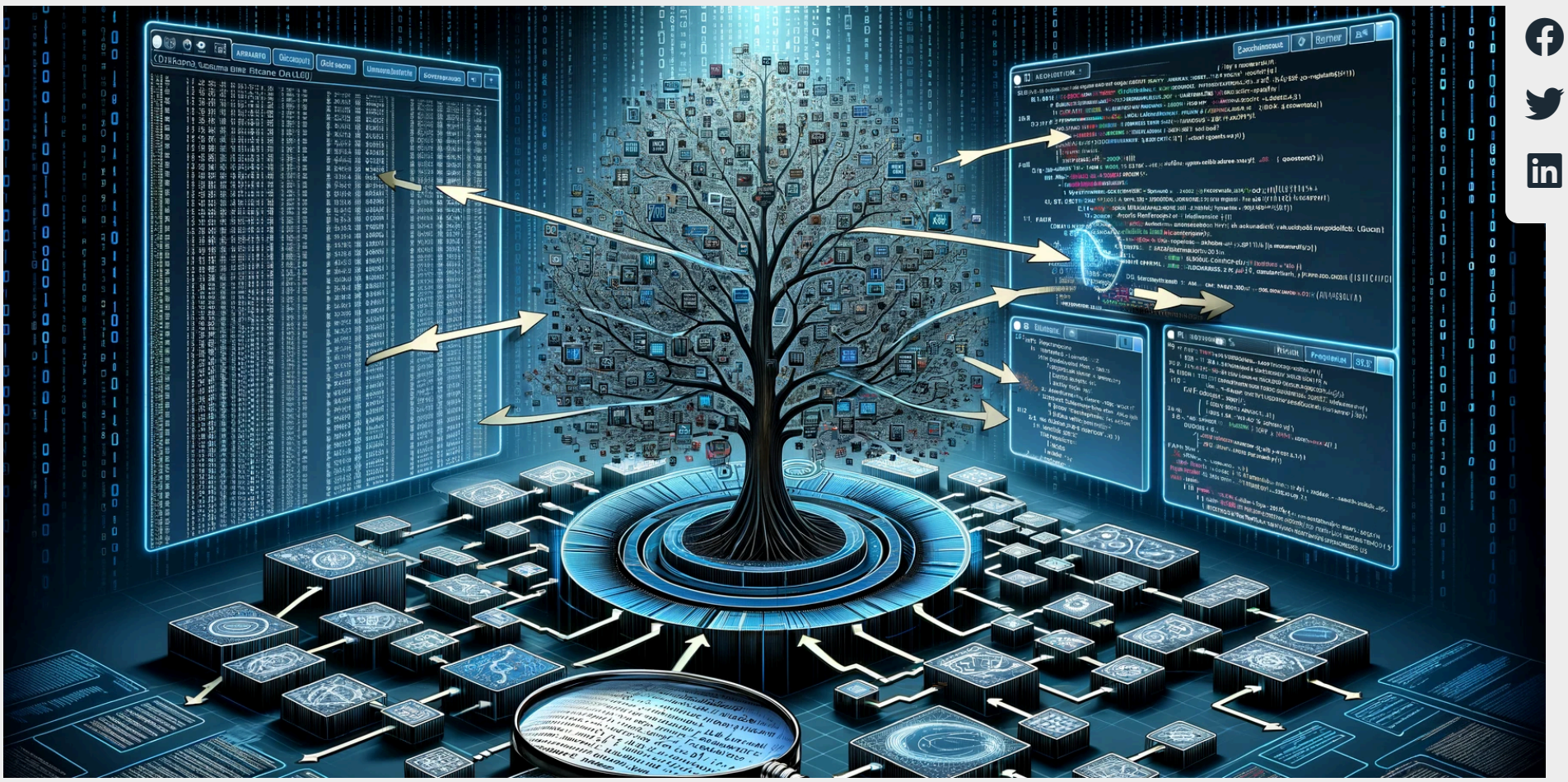
CTE Ricorsive: L'Eleganza della Semplicità in SQL – 3

15 Gennaio 2024



Introduzione

Come abbiamo già visto in altri Articoli, le Common Table Expressions (CTE) in SQL su IBM iSeries (AS/400), ma presenti anche in molte altre versioni su altri sistemi non IBM, rappresentano uno strumento potente e flessibile per creare query temporanee e riutilizzabili, semplificando e migliorando la leggibilità delle query SQL complesse..

Di seguito gli articoli della serie CTE (incluso questo corrente):

- iSeries Sql – CTE Common Table Expressions – definizioni e utilizzo
- CTE Ricorsive: L'Eleganza della Semplicità in SQL – 1
- CTE Ricorsive: L'Eleganza della Semplicità in SQL – 2
- CTE Ricorsive: L'Eleganza della Semplicità in SQL – 3
- Recursive CTE come concatenare i risultati su un'unica riga

In particolare le Common Table Expressions (CTE) **Ricorsive** sono un potente strumento in SQL che permette di **eseguire operazioni che, in assenza di questa funzionalità, richiederebbero lo sviluppo di programmi specifici o l'utilizzo di cicli complessi**.

In questo Articolo vedremo una ulteriore estensione dell'esempio **CTE Ricorsive: L'Eleganza della Semplicità in SQL – 1** e vedremo come un comando SQL con utilizzo di CTE Ricorsive può dare risultati che normalmente richiederebbero molte righe di codice se risolti con un approccio programmatico tradizionale.

Va detto che l'utilizzo delle CTE ricorsive non è banale e va affrontato con attenzione ma se viene capito l'approccio può dare soluzioni concise ed eleganti per risolvere problemi anche complessi.

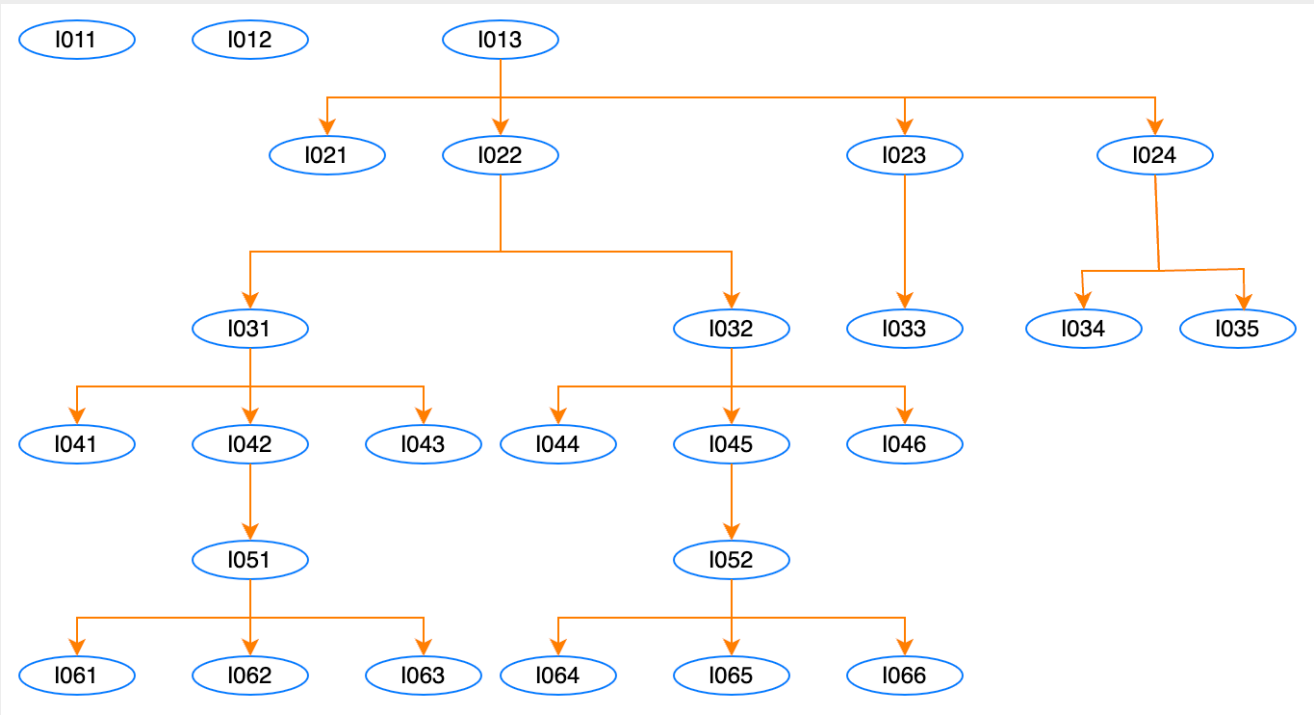
Descrizione Scenario e obiettivo

Immaginiamo uno schema di rete organizzato in una struttura ad albero, dove sono illustrati sia i nodi sia le relazioni di dipendenza che intercorrono tra di essi.

Questi nodi possono simboleggiare vari elementi, come i dipartimenti di un'azienda, i dipendenti di un'organizzazione, o i nodi di un circuito elettronico, dove sono delineate le relazioni di dipendenza tra i diversi processori, e così via.

Ipotizziamo di voler compilare un elenco di tutti i percorsi unici che partono dalla radice e raggiungono le foglie nell'albero di nodi, includendo anche quei nodi "isolati", ossia privi di dipendenze o collegamenti con altri nodi.

Di seguito uno schema della struttura ad albero:



In questa struttura sono presenti nodi "isolati", ovvero senza legami e nodi aventi uno o più legami.

Il risultato che si vuole ottenere è l'elenco di tutti i percorsi unici (che non siano inclusi in altri percorsi) che partono da una radice (nodo senza un "padre") e raggiungono le foglie terminali (nodi senza "figli") nell'albero di nodi, includendo anche quei nodi "isolati", ossia privi di dipendenze o collegamenti con altri nodi.

Query

```
WITH -- starting data
     TreeStructure (NodeID, Description, ParentNodeID) AS (
         VALUES  ('I011', 'Description_011',  NULL) , ('I012', 'Description_012',  NULL)
               , ('I013', 'Description_013',  NULL) , ('I021', 'Description_021', 'I013')
```

```
              , ('I022', 'Description_022', 'I013'), ('I023', 'Description_023', 'I013')
              , ('I024', 'Description_024', 'I013'), ('I031', 'Description_031', 'I022')
              , ('I032', 'Description_032', 'I022'), ('I041', 'Description_041', 'I031')
              , ('I042', 'Description_042', 'I031'), ('I043', 'Description_043', 'I031')
              , ('I044', 'Description_044', 'I032'), ('I045', 'Description_045', 'I032')
              , ('I046', 'Description_046', 'I032'), ('I051', 'Description_051', 'I042')
              , ('I052', 'Description_052', 'I046'), ('I061', 'Description_061', 'I051')
              , ('I062', 'Description_062', 'I051'), ('I063', 'Description_063', 'I051')
              , ('I064', 'Description_064', 'I052'), ('I065', 'Description_065', 'I052')
              , ('I066', 'Description_066', 'I052'), ('I033', 'Description_033', 'I023')
              , ('I034', 'Description_034', 'I024'), ('I035', 'Description_035', 'I024')
)
, Routes(NodeID, Path) AS (
    -- Parte ancorata: seleziona i nodi radice
    SELECT NodeID, CAST(NodeID AS VARCHAR(1024))
    FROM TreeStructure
    WHERE ParentNodeID IS NULL
    UNION ALL
    -- Parte ricorsiva: aggiungi il nodo corrente al Path
    SELECT nd.NodeID, p.Path concat '->' concat nd.NodeID
    FROM TreeStructure nd
    JOIN Routes p ON nd.ParentNodeID = p.NodeID
)
, Routes2(NodeID, Path) AS (
    SELECT NodeID, SUBSTR(Path, 1, length(Path)-6)
      FROM Routes
      WHERE LENGTH(Path) > 6
    UNION ALL
    SELECT NodeID, Path
      FROM Routes
      WHERE LENGTH(Path) <= 6
)
, Routes3(NodeID, Path) AS (
    SELECT * FROM Routes P1
     WHERE NOT EXISTS (
           SELECT 1 from Routes2 P2
            WHERE P2.Path = P1.Path)
      ORDER BY 2
) --  SELECT * FROM Routes3;
 SELECT * FROM Routes3
  UNION ALL
 SELECT NodeID, NodeID
   FROM TreeStructure N1
  WHERE NOT EXISTS (
            SELECT 1 FROM TreeStructure N2
              WHERE N1.NodeID = N2.ParentNodeID)
    AND N1.ParentNodeID IS NULL
   ORDER BY 2;
```

Risultato della Query

ID_NODO	PERCORSO
I011	I011
I012	I012
I021	I013->I021
I041	I013->I022->I031->I041
I061	I013->I022->I031->I042->I051->I061
I062	I013->I022->I031->I042->I051->I062
I063	I013->I022->I031->I042->I051->I063
I043	I013->I022->I031->I043
I044	I013->I022->I032->I044
I045	I013->I022->I032->I045
I064	I013->I022->I032->I046->I052->I064
I065	I013->I022->I032->I046->I052->I065
I066	I013->I022->I032->I046->I052->I066
I033	I013->I023->I033
I034	I013->I024->I034
I035	I013->I024->I035

Descrizione dettagliata della query

La query SQL descritta è un esempio di una Common Table Expression (CTE) ricorsiva utilizzata per elaborare una struttura ad albero. La query è suddivisa in più parti, ognuna delle quali costruisce sopra la precedente per arrivare al risultato finale.

1) Definizione della Struttura dell'Albero (TreeStructure):

- Questa CTE iniziale, **TreeStructure**, definisce i dati di partenza, rappresentando l'albero dei nodi. Ogni nodo è definito da un **NodeID** univoco, una **Description** e un **ParentNodeID** che indica il nodo genitore (se esiste; se **NULL**, il nodo è una radice).

2) Generazione dei Percorsi (Routes):

- La CTE **Routes** inizia con la parte "ancorata" che seleziona tutti i nodi radice (i nodi senza un nodo genitore).

- Poi, attraverso la parte "ricorsiva", la query si espande aggiungendo progressivamente i nodi figli ai percorsi, concatenando il **NodeID** corrente al percorso (**Path**) del nodo genitore.

3) Pulizia dei Percorsi (Routes2)

- La CTE **Routes2** è una tabella intermedia che serve per individuare eventuali percorsi che sono inclusi in altri percorsi (sotto percorsi). In pratica per poter individuare i sotto-percorsi viene creata questa tabella intermedia come copia della tabella Routes nella quale viene modificata la colonna Path rimuovendo l'ultimo nodo (creando così sotto-percorsi per confrontarli).

4) Selezione dei Percorsi Unici (Routes3)

- **Routes3** seleziona i percorsi finali escludendo quelli che sono sotto-percorsi di altri percorsi già esistenti, assicurando che ogni percorso sia unico e non sia incluso in un altro percorso più lungo.

5) Risultato Finale

- Infine, la query seleziona tutti i percorsi unici dall'ultima CTE (**Routes3**).
- Aggiunge anche i nodi "isolati", che sono nodi radice senza figli, utilizzando un **UNION ALL** per unire questi nodi alla lista dei percorsi. I nodi isolati sono identificati confrontando **TreeStructure** con se stessa per trovare nodi che non sono genitori di altri nodi (cioè, non hanno figli) e che non hanno un nodo genitore (sono nodi radice).
- La parte finale della query ordina tutti i risultati basandosi sulla colonna **Path** per presentare l'elenco in un ordine leggibile.

In sintesi, la query è un modo sofisticato di trasformare una struttura ad albero rappresentata in una tabella di database in un elenco di percorsi unici da ogni nodo radice a ogni nodo foglia, includendo anche nodi isolati. Questa tecnica è spesso utilizzata in database che supportano CTE ricorsive per manipolare dati gerarchici o ricavare informazioni da strutture ad albero complesse.

Per eseguire lo script occorre assicurarsi che la sintassi specifica (**concat**, **SUBSTR**, **LENGTH**) sia supportata dal DBMS.

IBM iSeries utilizza una sintassi leggermente diversa da altri sistemi SQL per alcune funzioni. Come sempre tenere presente che l'implementazione e le funzionalità effettive potrebbero variare a seconda della versione del sistema operativo iSeries e del database DB2. È sempre buona norma fare riferimento alla documentazione ufficiale IBM per avere informazioni più precise e dettagliate.

#CTE Common Table Expressions | #Tree Structure | #SQL | #CTE Ricorsive

← PRECEDENTE
CTE Ricorsive: L'Eleganza della Semplicità in SQL – 2

SEGUENTE →
Recursive CTE come concatenare i risultati su un'unica riga

Articoli simili

CATEGORIE BLOG

Data Science

AI Artificial Intelligence

SkillUP

As400 Architettura

Gestire AS400 con SQL

Gestione approfondita Risorse Sistema

Spiccioli di tecnica

Windows

CERCA TESTO IN POST / TAGS

Cerca ...

SkillUp Via Marco Polo, 139 55049 Viareggio (LU)
Via Lucchese, 84 50019 Sesto Fiorentino (FI)
Tel: 055319212 - info@SkillUp.cloud

LinkedIn

Marco Del Corto
Founder/CEO Genesi srl
Visualizza profilo