Rekurzivní dotazy v SQL

Úvod

Rekurzivní SQL dotaz je takový dotaz, který ve své definici používá sebe sama. Takovýto dotaz je vhodný pro procházení dat s hierarchickou strukturou (záznamy mají nadřazený záznam).

CTE

Common Table Expression - "Obecný tabulkový výraz"

Zaveden Microsoftem v rámci SQL Server 2005

- V současnosti nejrozšířenější nástroj k řešení rekurzivních dotazů

Tabulka určená k ukládaní mezivýsledků v rámci složitějších vyhledávacích dotazů

- Předem vymezená životnost - pouze na první výraz typu SELECT, INSERT, UPDATE nebo DELETE následující po jeho vytvoření

Rekurzivní CTE v PSQL

Syntaxe:

```
WITH RECURSIVE [název CTE] AS
(
        [ukotvovací výraz]
    UNION [ALL]
        [rekurzivní výraz]
)
[výraz pracující s CTE];
```

- Modifikátor RECURSIVE je pro správné fungování nezbytný
- Spojení ukotvovacího a rekurzivního výrazu:
    - UNION - ukládá každou n-tici pouze jednou
    - UNION ALL - ukládá i duplikáty n-tic

Praktické příklady

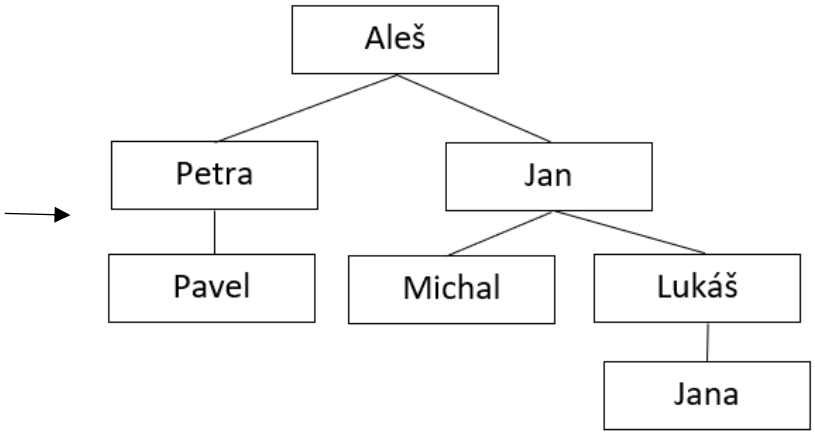
Hierarchie ve firmě

Mějme tabulku obsahující záznamy o zaměstnancích firmy.

Každý zaměstnanec této firmy má jméno a jednoho přímého nadřízeného (ketrý je také zaměstnanec a jako takový má také nadřízeného). Jediný zaměstnanec ketrý nemá nadřízeného je Aleš a bude tedy kořenem stromové hierarchie této firmy.

Takováto firemní struktura by šla znázornit stromem takto:

ID	JMENO	BOSS_ID
1	Aleš	NULL
2	Jan	1
3	Lukáš	2
4	Pavel	6
5	Michal	2
6	Petra	1
7	Jana	3



Otázky:

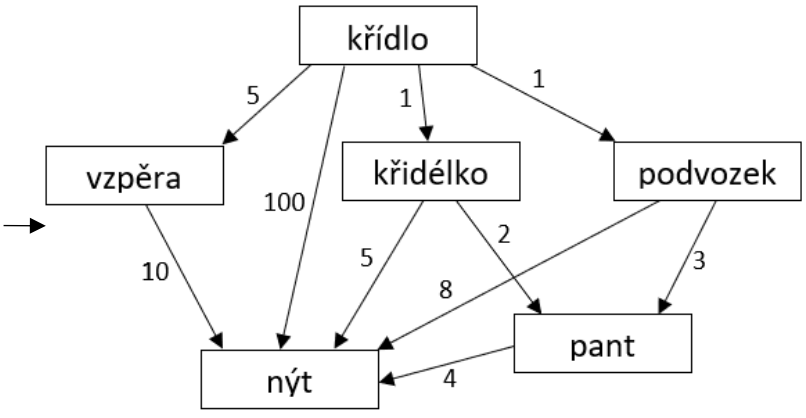
1. Kteří zaměstnanci jsou podřízení Jana?
2. Je Jan nadřízený Pavla?
3. V jaké úrovni firemní struktury se nachází Jana?

Komponenty a podkomponenty

Mějme tabulku se seznamem komponent a s počty součástek potřebných pro výrobu těchto komponent (součástka je opět komponenta)

Vztah mezi komponentou a její součástí můžeme opět znázornit grafem:

KOMPONENTA	SOUCAST	POCET
křídlo	vzpěra	5
křídlo	křidélko	1
křídlo	podvozek	1
křídlo	nýt	100
vzpěra	nýt	10
křidélko	pant	2
křidélko	nýt	5
podvozek	pant	3
podvozek	nýt	8
pant	nýt	4



Otázky:

1. Seznam všech komponent potřebných na výrobu křídla (včetně počtů)?

Řešení otázek v SQL

1. Kteří zaměstnanci jsou podřízení Jana?

```sql
WITH RECURSIVE podrizeni (id, jmeno, boss_id) AS (
    SELECT id,jmeno,boss_id
    FROM firma
    WHERE jmeno = 'Jan'
    UNION ALL
    SELECT f.id,f.jmeno,f.boss_id
    FROM firma f, podrizeni p
    WHERE f.boss_id = p.id
)
SELECT id,jmeno FROM podrizeni WHERE jmeno != 'Jan';
```

Výsledek:

ID	JMENO
3	Lukáš
5	Michal
7	Jana

2. Je Jan nadřízený Pavla?

```sql
WITH RECURSIVE podrizeni (id,jmeno,boss_id) AS(
    SELECT id, jmeno, boss_id
    FROM firma
    WHERE jmeno = 'Jan'
    UNION ALL
    SELECT f.id, f.jmeno, f.boss_id
    FROM firma f, podrizeni p
    WHERE f.boss_id = p.id
)
SELECT EXISTS(SELECT id, jmeno FROM podrizeni WHERE jmeno = 'Pavel') AS je_podrizeny;
```

Výsledek:

JE_PODRIZENY
false

3. V jaké úrovni firemní struktury se nachází Jana?

```sql
WITH RECURSIVE podrizeni (id, jmeno, boss_id, uroven) AS(
    SELECT id, jmeno, boss_id, 0 as uroven
    FROM firma
    WHERE boss_id IS NULL
    UNION ALL
    SELECT f.id, f.jmeno, f.boss_id, uroven + 1
    FROM firma f, podrizeni p
    WHERE f.boss_id = p.id
)
SELECT jmeno, uroven FROM podrizeni WHERE jmeno = 'Jana';
```

Výsledek:

JMENO	UROVEN
Jana	3

1. Seznam všech komponent potřebných na výrobu křídla (včetně počtů)?

```
WITH RECURSIVE Casti (soucast, pocet) AS(
    SELECT soucast, pocet
    FROM Komponenty
    WHERE komponenta = 'křídlo'
    UNION ALL
    SELECT k.soucast, c.pocet * k.pocet
    FROM Casti c, Komponenty k
    WHERE c.soucast = k.komponenta
)
SELECT soucast, sum(pocet) AS pocet FROM Casti GROUP BY soucast;
```

Vysledek:

SOUCAST	POCET
pant	5
nýt	183
křidélko	1
podvozek	1
vzpěra	5

Zacyklení rekurze

Pokud se při průchodu nasí strukturou tvoří cyklus, nebo není nijak definovaný konec rekurze, rekurzivní dotaz se nemusí zastavit. V takovém případě musíme konec rekurze zajistit.

Příklad:

```
WITH RECURSIVE cisla (cislo) AS(
    SELECT 1 AS cislo
    UNION ALL
    SELECT cislo + 1 FROM cisla
)
SELECT cislo FROM cisla;
```

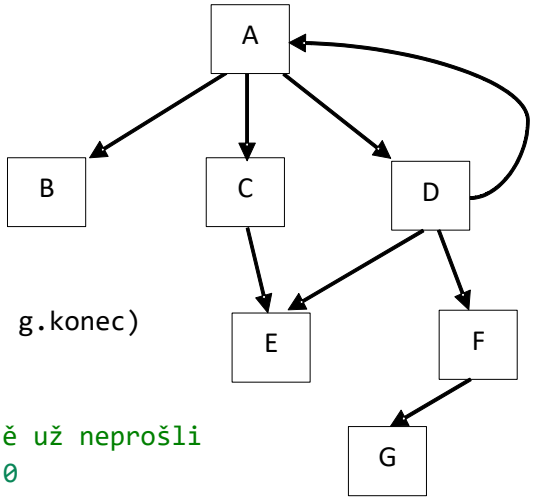
Aby se dotaz nezacyklil, můžeme dotaz omezit na několik prvních výsledků pomocí výrazu LIMIT:

```
WITH RECURSIVE cisla (cislo) AS(
    SELECT 1 AS cislo
    UNION ALL
    SELECT cislo + 1 FROM cisla
)
SELECT cislo FROM cisla LIMIT 20;
```

Teď nám dotaz vrátí jenom 20 prvních řádků. Další možností je sledovat hloubku rekurze přímo uvnitř dotazu:

```
WITH RECURSIVE cisla (cislo) AS(
    SELECT 1 AS cislo
    UNION ALL
    SELECT cislo + 1 FROM cisla
    WHERE cislo < 20
)
SELECT cislo FROM cisla;
```

Při průchodu grafem si můžeme pamatovat již navštívené uzly a ty potom vynechat. V následujícím příkladu se budeme snažit najít možné cesty z bodu **A** do ostatních bodů orientovaného grafu tak, aby každá cesta prošla každým vrcholem nanejvýš jednou.



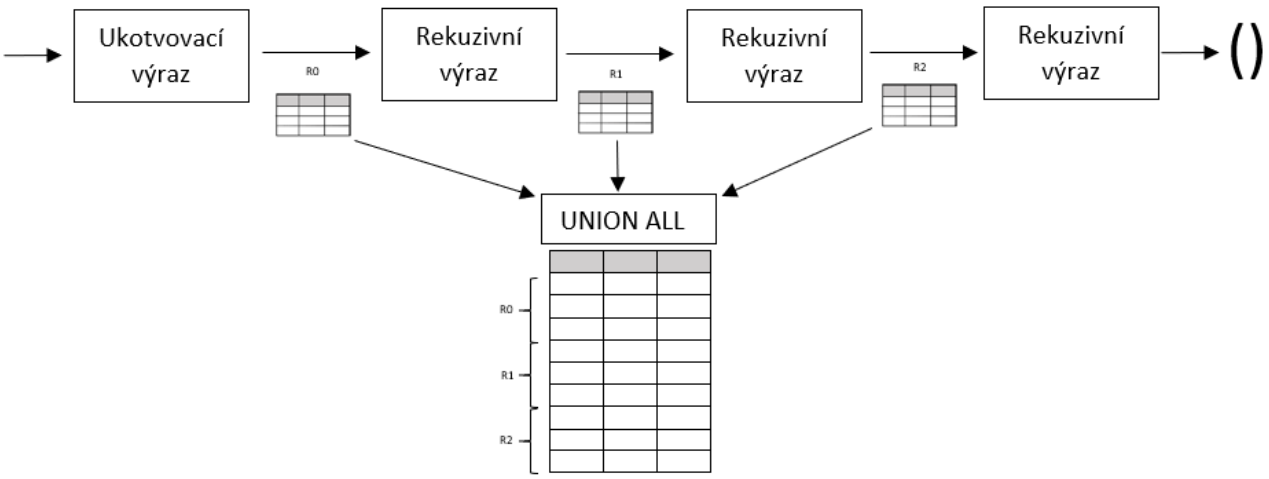

```
WITH RECURSIVE cesty (konec, delka, cesta) AS(
    SELECT DISTINCT start, 0, start
    FROM graf
    WHERE start = 'A'
    UNION ALL
    SELECT g.konec, c.delka + 1, CONCAT(c.cesta, '>', g.konec)
    FROM graf g, cesty c
    WHERE c.konec = g.start
    -- tady kontrolujeme, jestli jsme vrcholem v cestě už neprošli
    AND POSITION(CONCAT(g.konec , '>') IN c.cesta) = 0
)
SELECT konec, delka, cesta
FROM cesty;
```

Překročení velikosti dat

Pokud bychom v předešlém příkladu nastavili délky řetězce na příliš malou (například typ varchar(3)), celá cesta se neuloží. Tento problém můžeme vyřešit přetypováním pomocí funkce ČÁST().

```
…
(SELECT DISTINCT start, 0, CAST(start AS varchar(20))
 -- přetypujeme proměnnou v inicializačním poddotazu
…
SELECT g.konec, c.delka + 1,
CAST(CONCAT(c.cesta, '>', g.konec) AS varchar(20))
-- přetypujeme proměnnou v rekurzivní části
…
```

Kroky rekuzivního výpočtu



Řešení na platformě ORACLE

Podpora CTE (včetně rekurzivních) od verze Oracle Database 11g Release 2

Klauzule START WITH, CONNECT BY

Syntaxe:

```
[dotaz]
START WITH
     [selekce “kořenových” n-tic]
CONNECT BY [NOCYCLE] PRIOR
     [atributy určující relaci rodič-potomek];
```

- klauzuli START WITH možno vynechat, pokud mají být kořenem všechny n-tice původní relace
- Modifikátor NOCYCLE nutný v případě, kdy se v relaci vyskytují cykly
- Pseudoatributy (uživatel je může číst, ale ne měnit) spjaté s rekurzivními dotazy::
  - CONNECT_BY_ISLEAF - n-tice bez potomku mají hodnotu 1, ostatni 0
  - CONNECT_BY_ISCYCLE - cyklicke n-tice (předek je zároveň potomkem) nabývají hodnotu 1, ostatní 0

Zdroje:

https://www.ksi.mff.cuni.cz/~pokorny/dj-new/DJ2-3-rekurze.pdf

https://www.postgresql.org/docs/13/queries-with.html

https://www.oradev.com/connect_by.jsp

https://medium.com/swlh/recursion-in-sql-explained-graphically-679f6a0f143b