SQL Common Table Expression

Dominique Dumont

Qui suis-je ?

- Dominique Dumont (@dod38fr)
- Debian Developer depuis 2011
- Contributeur Open-Source depuis 1996
- Devops freelance depuis 2020

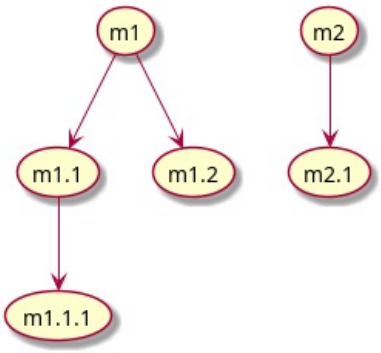
C'est quoi une CTE ?

- Common Table Expressions
- Fait partie du SQL, langage d'interrogation de base de donnée relationelle
- permet de rentre le SQL plus lisible
- permet de définir des tables temporaires exploitées par des requêtes
- supporte la récursivité

Comment ça fonctionne ?

Exemple arbre: des messages

- hiérarchie de messages
- les msg_id représentent un chemin (*path*) pour faciliter la compréhension



Exemple: la table des messages

```sql
drop table if exists message;
create table message (
  -- clef privée (surrogate key)
  id integer primary key asc autoincrement,
  -- définie une structure en arbre
  parent_id integer references message (id) on delete set null,
  -- clef métier (natural key)
  message_key text not null unique,
  subject text
);
```

les messages dans la table

```sql
select * from message
```

id	parent_id	message_key	subject
1		m1	bla-bla
2	1	m1.1	Re: bla-bla
3	2	m1.1.1	Re: Re: bla-bla
4	1	m1.2	Re: bla-bla
5		m2	meh
6	5	m2.1	Re: meh

Exemple: Trouver un parent

Ici, pas besoin de CTE, mais la logique de la requête est à l'envers:

```sql
-- 2. trouver le message_key du parent
select message_key, subject from message
where id == (
  -- 1. trouver l'id du parent
  select parent_id from message where message_key = "m1.1"
)
```

message_key	subject
m1	bla-bla

Exemple: Trouver un parent avec une CTE

- but: éviter la sous-requête
- construit une table temporaire utilisable dans la requête suivante

```sql
-- trouver l'id du parent
with parent(id) as (
  select parent_id from message where message_key = "m1.1.1"
)
-- trouver le message_key du parent
select message_key, subject from message,parent
where message.id == parent.id
```

message_key	subject
m1.1	Re: bla-bla

Problème: trouver le premier message d'un fil

- besoin de récursion
- CTE simple non suffisante

CTE récursive - 1ere étape de la récursion

- paramètres nécessaire: `parent_id`
- `id` et `subject` sont présent pour faciliter la compréhension

```
with recursive tmp_msg(id, message_key, subject, parent_id) as (
  select id, message_key, subject, parent_id from message
  where message_key = "m1.1.1"
)
select * from tmp_msg
```

id	message_key	subject	parent_id
3	m1.1.1	Re: Re: bla-bla	2

CTE récursive - mise en place de la récursion

- récursion: la 2e requête utilise la table définie avec *with*

```
with recursive tmp_msg(id, message_key, subject, parent_id) as (
-- étape 1
  select id, message_key, subject, parent_id from message
  where message_key = "m1.1.1"
  union all -- pour coller les autres étapes
  select a.id, a.message_key, a.subject, a.parent_id from message as
        a
  join tmp_msg where a.id = tmp_msg.parent_id -- □ la récursion est
        là
  limit 10 -- filet anti crash pour le debug
)
select * from tmp_msg
```

id	message_key	subject	parent_id
3	m1.1.1	Re: Re: bla-bla	2
2	m1.1	Re: bla-bla	1
1	m1	bla-bla	

CTE récursive - sous le capot

- ajout de la variable `count` pour voir les étapes
- chaque étape voit le résultat de l'étape précédente

```
with recursive tmp_msg(count, id, message_key, subject, parent_id)
        as (
  -- étape 1
  select 1, id, message_key, subject, parent_id from message
    where message_key = "m1.1.1"
  union all
  select count+1, a.id, a.message_key, a.subject, a.parent_id from
        message as a
  join tmp_msg on a.id = tmp_msg.parent_id
)
select * from tmp_msg
```

count	id	message_key	subject	parent_id
1	3	m1.1.1	Re: Re: bla-bla	2
2	2	m1.1	Re: bla-bla	1
3	1	m1	bla-bla	

CTE récursive - récupération du résultat:

- enlever toutes les variables inutiles

```
with recursive tmp_msg(id, message_key, subject, parent_id) as (
  select id, message_key, subject, parent_id from message where
        message_key = "m1.1.1"
  union all
  select a.id, a.message_key, a.subject, a.parent_id from message as
        a
  join tmp_msg on a.id = tmp_msg.parent_id
)
select message_key, subject from tmp_msg
where parent_id isnull
```

message_key	subject
m1	bla-bla

Autre exemple: tous les fils

```
with recursive tmp_msg(id) as (
  select id from message where message_key = "m1"
  union all
  select a.id from message as a
  join tmp_msg where a.parent_id = tmp_msg.id
)
select message_key, subject from tmp_msg
join message on message.id == tmp_msg.id
```

message_key	subject
m1	bla-bla
m1.1	Re: bla-bla
m1.1.1	Re: Re: bla-bla
m1.2	Re: bla-bla

Conclusion

- les CTEs peuvent clarifier le code SQL
- Gain de performance en évitant les allers et retours entre le service et la DB
- on peut définir plusieurs tables dans une CTE
- attention:
  - au nommage des tables temporaires
  - aux différences possibles entre les databases (MySQL, Postgress...)