



Power Saturday

SQL Server 2019, de l'intelligence sous le capot
Arian Papillon & Frédéric Brouard

Les speakers

- Arian Papillon
a.papillon@datafly.fr
<http://blog.datafly.pro>
<http://youtube.datafly.fr>
<http://mssql.fr>
- Frédéric Brouard
sqlpro@sqlspot.com
<https://sqlpro.developpez.com>
<http://mssqlserver.fr>



Exécution d'une requête SQL

```
SELECT INSEE_COM, NOM_COM,
       POPULATION
FROM   S_GEO.COMMUNE
WHERE  NOM_DEPT = 'VAR'
```

ANALYSE

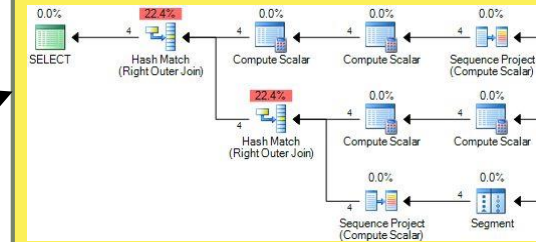
OPTIMISATION

EXÉCUTION

Méthode
d'accès

	INSEE_COM	NOM_COM	POPULATION
1	83108	LA ROQUEBRUSSANNE	2526
2	83115	SAINTE-MAXIME	13835
3	83149	VILLECROZE	1366
4	83053	EVENOS	2141
5	83005	ARTIGNOSC-SUR-VERDON	319
6	83007	AUPS	2122
7	83107	ROQUEBRUNE-SUR-ARGENS	12344
8	83046	COTIGNAC	2292
9	83119	SAINT-TROPEZ	4402
10	83029	CALLIAN	3385
11	83032	CARCES	3406
12	83000	DIJET-SUR-ARGENS	7116

CACHE DES PLANS



CACHE DES DATA

C...	COD...	NOM_CHF	X_CHF_LIEU	Y_CHF_LIEU	X_CEN
26	003	AMBARES-ET-L...	424980	6431138	42668
11	159	MONTS	521813	6688702	52722
11	143	MUZILLAC	288059	6731280	29788
21	551	THIVIERS	537451	6481936	54100
12	257	EVRECY	444241	6894566	45131
15	115	SAINTE-MAXIME	995208	6252338	98813
22	268	DESVRES	617462	7064045	61388
23	263	ORCINES	700959	6520316	69316
06			0	0	43235
08			0	0	78441
18	187	PLERIN	274342	6841599	27167
09	033	FONTENAY-SOU...	661629	6861422	66136
15			0	0	76436
21			0	0	66875
17	516	VITTEL	919086	6793139	91672
99	194	NEVERS	711912	6654718	71218
08			0	0	57467
03	033	BAR-SUR-AUBE	826691	6793901	82645
02	049	BRESSUIRE	434221	6643640	43582
18	070	SAINT-OUEN	651102	6867647	65027
99	300	LONS-LE-SAUNI...	895198	6622537	89553
10	410	BECHBONNIEU	576395	6290630	58351
04	106	BOUZONVILLE	957074	6916002	95484
10			0	0	10376
08	075	CHAMALIERES	705209	6519339	70370
08			0	0	95722
40	009	VILLENEUVE-D'...	710225	7058427	71083
02	056	BRUNSTATT	1023906	6744748	10316
06	106	COURSAN	704796	6237043	71098
10	128	GRAMAT	599131	6409684	59530
17	246	LES PONTS-DE...	434324	6708618	43954

Exécution d'une requête SQL

```
SELECT INSEE_COM, NOM_COM,
       POPULATION
FROM   S_GEO.COMMUNE
WHERE  NOM_DEPT = 'VAR'
```

ANALYSE

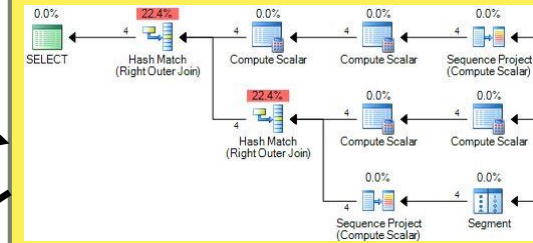
OPTIMISATION

EXÉCUTION

Méthode
d'accès

	INSEE_COM	NOM_COM	POPULATION
1	83108	LA ROQUEBRUSSANNE	2526
2	83115	SAINTE-MAXIME	13835
3	83149	VILLECROZE	1366
4	83053	EVENOS	2141
5	83005	ARTIGNOSC-SUR-VERDON	319
6	83007	AUPS	2122
7	83107	ROQUEBRUNE-SUR-ARGENS	12344
8	83046	COTIGNAC	2292
9	83119	SAINT-TROPEZ	4402
10	83029	CALLIAN	3385
11	83032	CARCES	3406
12	83000	DIJON-SUR-ARGENS	7116

CACHE DES PLANS



CACHE DES DONNÉES

C...	COD...	NOM_CHF	X_CHF_LIEU	Y_CHF_LIEU	X_CEN
26	003	AMBARES-ET-L...	424980	6431138	42668
11	159	MONTS	521813	6688702	52722
11	143	MUZILLAC	288059	6731280	29788
21	551	THIVIERS	537451	6481936	54100
12	257	EVRECY	444241	6894566	45131
15	115	SAINTE-MAXIME	995208	6252338	98813
22	268	DESVRES	617462	7064045	61388
23	263	ORCINES	700959	6520316	69316
06			0	0	43235
08			0	0	78441
18	187	PLERIN	274342	6841599	27167
09	033	FONTENAY-SOU...	661629	6861422	66136
15			0	0	76436
21			0	0	66875
17	516	VITTEL	919086	6793139	91672
99	194	NEVERS	711912	6654718	71218
08			0	0	57467
03	033	BAR-SUR-AUBE	826691	6793901	82645
02	049	BRESSUIRE	434221	6643640	43582
18	070	SAINT-OUEN	651102	6867647	65027
99	300	LONS-LE-SAUNI...	895198	6622537	89553
10	410	BECHBONNIEU	576395	6290630	58351
04	106	BOUZONVILLE	957074	6916002	95484
10			0	0	10376
08	075	CHAMALIERES	705209	6519339	70370
08			0	0	95722
40	009	VILLENEUVE-D'...	710225	7058427	71083
02	056	BRUNSTATT	1023906	6744748	10316
06	106	COURSAN	704796	6237043	71098
10	128	GRAMAT	599131	6409684	59530
17	246	LES PONTS-DE...	434324	6708618	43954

Optimiseur intelligent ? Pas sans DBA...

- L'optimiseur adapte son plan d'exécution en fonction des données, mais il fait parfois des erreurs !
 - Mauvaise estimation -> mauvais plan et/ou mauvaise allocation de mémoire -> mauvaises performances
- Un « tuning » manuel peu s'avérer nécessaire
 - Guides de plan, query store, statistiques, index, réécriture...
- Le DBA n'est pas toujours là... comment faire pour que l'optimiseur gère mieux ses erreurs tout seul ?

Optimisation automatique

- Depuis SQL Server 2017, Microsoft travaille à améliorer le moteur avec des fonctionnalités d'optimisation automatique :
 - Intelligent query processing
 - Automatic tuning



Fonctionnement de l'optimiseur (jusqu'ici)

- Compile son plan d'exécution, puis l'exécute après
 - Prend ses décisions en fonction de l'estimation des cardinalités (entre autres éléments), **avant l'exécution**
 - Inconvénient : si l'estimation est inexacte, ses choix peuvent être mauvais, la mémoire nécessaire mal estimée, etc...
- Stocke le plan en cache pour réutilisation
 - Si le plan est mauvais, répète la même erreur !
 - Problème du « parameter sniffing »

Autres problématiques d'optimisation courantes

- On connaît des cas typiques où l'optimiseur peut être inefficace
 - Utilisation de fonctions scalaires,
 - Utilisation de fonctions tables à instructions multiples
 - Utilisations de variables table
 - Régression de plans d'exécution
 - Index manquants
 - ...
- Optimisation sur de fortes volumétries
 - Travailler avec plus de lignes : mode « row » vs mode « batch »

Première piste : décider lors de l'exécution

- Jointures adaptatives (2017)
- Exécution « entrelacée » pour fonctions tables à instructions multiples (2017)
- Compilation différée de variable table (2019)

JUST DO IT.

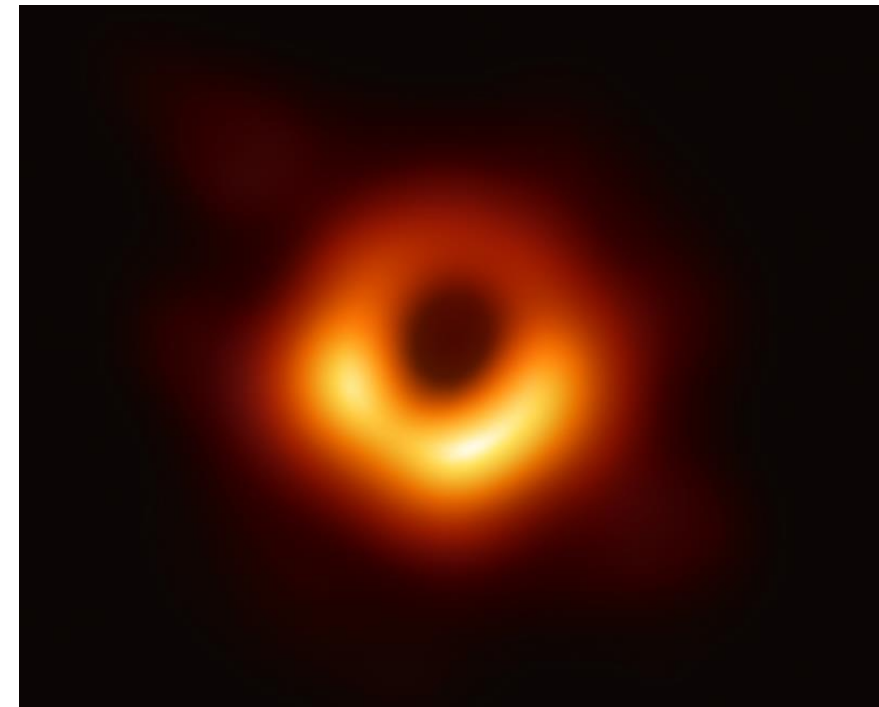


Jointures adaptatives (SQL 2017)

- Choix différé du type de jointure
 - *Hash match* (rapprochement par hachage) ou *nested loop* (boucles imbriquées)
 - Le choix est reporté après que la première entrée a été balayée
 - Opérateur de jointure spécifique (*adaptive join*)
- Compatibility level ≥ 140 : SQL Server 2017 minimum
- Edition Enterprise / Developer
- Mode batch seulement

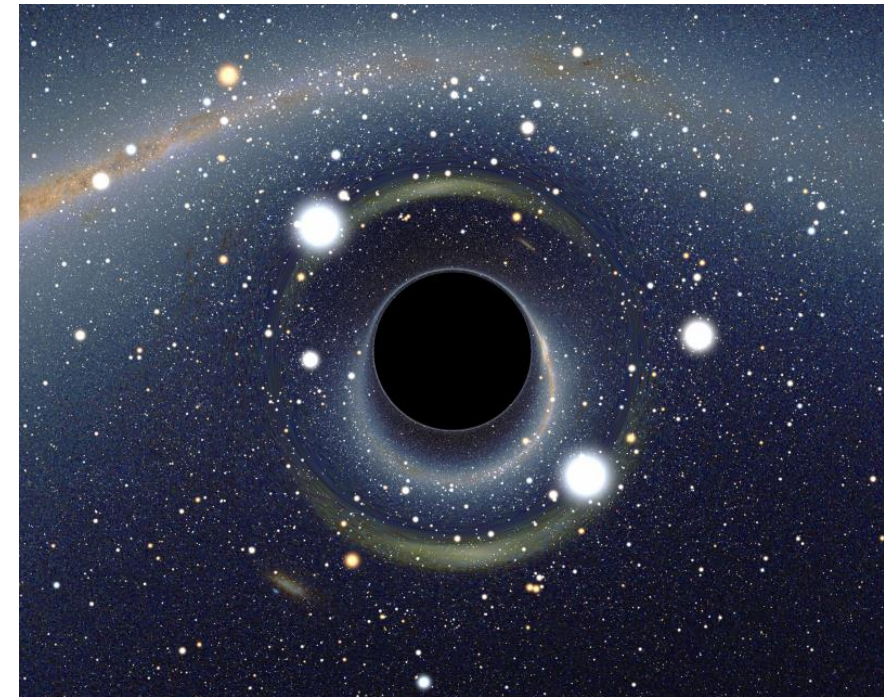
Exécution « entrelacée » des fonctions table (2017)

- Les fonctions tables à instructions multiples ne sont plus un trou noir
 - SQL Server 2014 : cardinalité estimée à 100 lignes (autrefois **1** dans les versions antérieures !)
- Fonctionnement : met en pause l'optimisation, estime la cardinalité de la fonction (sous-arbre du plan d'exécution), puis continue l'optimisation



Compilation différée variables tables (2019)

- Les variables tables ne sont plus un trou noir
 - Jusqu'à SQL 2017 : cardinalité estimée à 1
- La compilation du plan de requête est reportée au moment de l'exécution
 - Vraies estimations de cardinalités
 - Similaire à l'utilisation d'une table temporaire
 - Mais pas de statistiques...

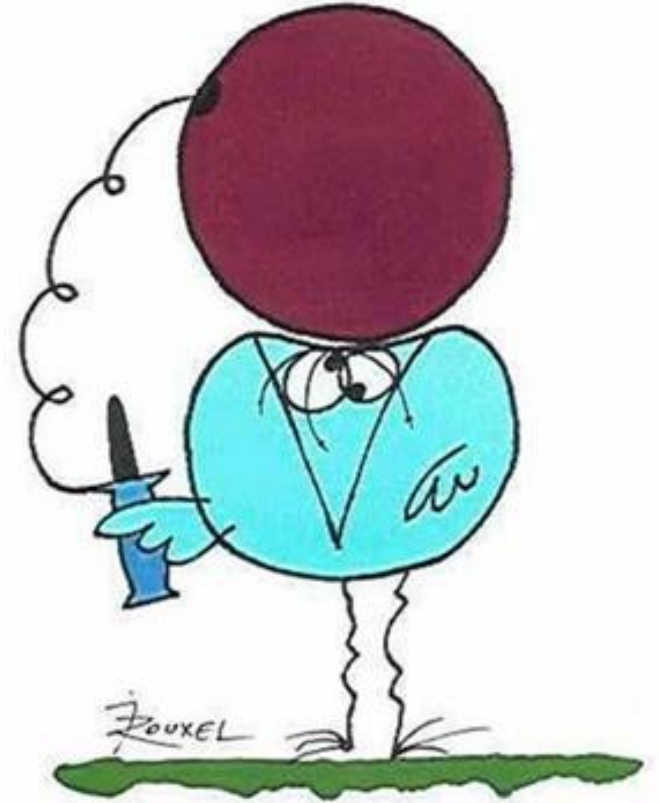


démo

2ème piste : s'inspirer du passé

- Feedback d'allocation mémoire (2017 + 2019)
 - En mode batch (SQL 2017) et en mode row (SQL 2019)
- Correction de plan automatique (2017 & Azure)

Les devises Shadok



EN ESSAYANT CONTINUUELLEMENT
ON FINIT PAR RÉUSSIR. DONC:
PLUS ÇA RATE, PLUS ON A
DE CHANCES QUE ÇA MARCHE.

Feedback d'allocation mémoire

- Réajuste la mémoire : modifie le plan d'exécution mis en cache pour les exécutions suivantes
- Déclenchement du recalcul
 - Surallocation : si mémoire allouée > (2 x mémoire utilisée) et (mémoire > 1 MB)
 - Sous-allocation : recalcule si débordement (spill) sur disque (tempdb)
- Mode batch seulement avec SQL 2017, maintenant aussi en mode row depuis SQL 2019 !
- Edition Enterprise / Developper

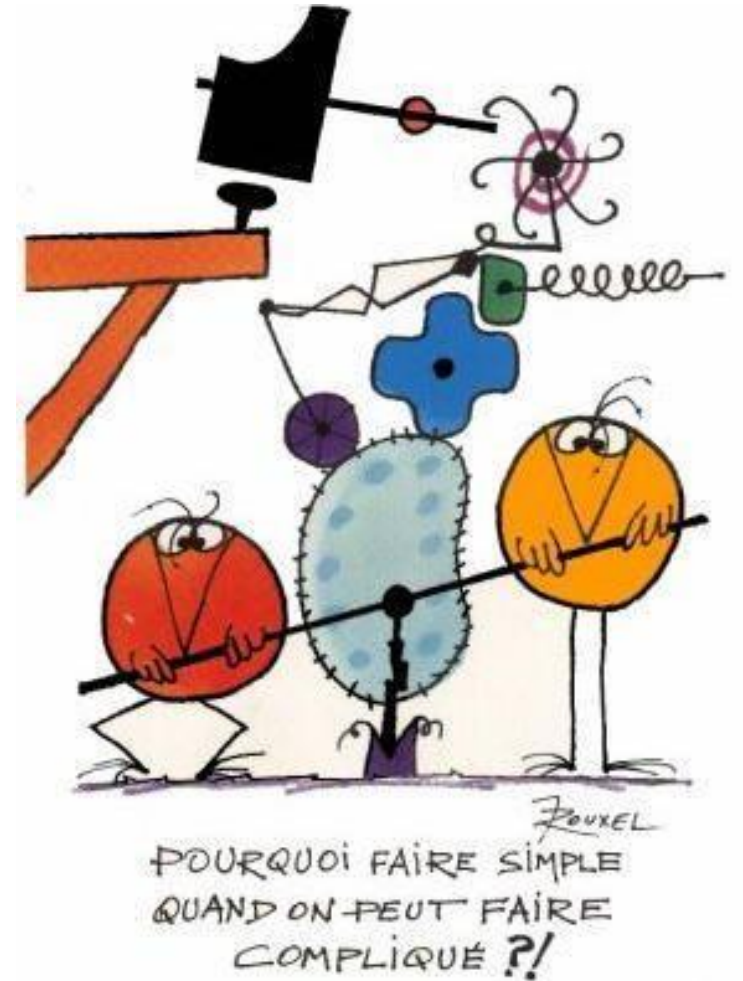
Correction de plan automatique (2017)

- S'appuie sur le Query Store
- Si régression, forçage automatique du « last known good plan », si :
 - Moins d'erreurs
 - Gain CPU estimé global > 10 seconds
 - Vérifie si le plan forcé est réellement meilleur
- *On premise*, édition Enterprise / Developer
 - Désactivé par défaut, activable au niveau base de données
 - `ALTER DATABASE <yourDB> SET AUTOMATIC_TUNING (FORCE_LAST_GOOD_PLAN = ON);`
- Azure SQL DB, pour tous les niveaux de service
 - Activé par défaut, configurable depuis le portail, au niveau serveur et base de données ou `ALTER DATABASE`

démo

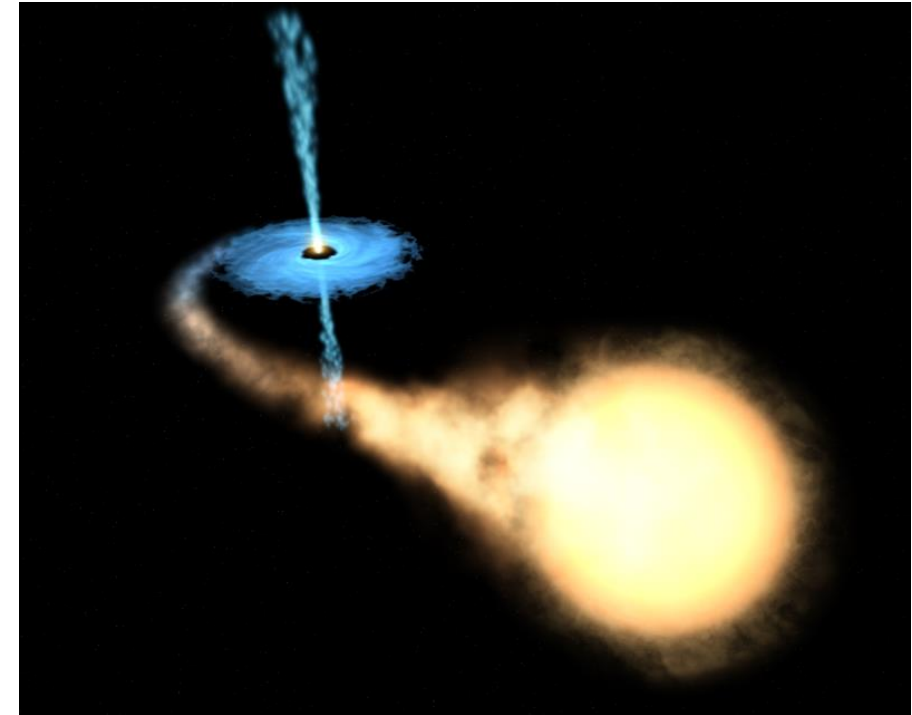
3^{ème} piste : optimiser les fonctions scalaires

- Incorporation des fonctions scalaires (2019)



Fonctions utilisateur (UDF) scalaires en TSQL

- Avantages : programmation modulaire
- Inconvénients : mauvaises performances :
 - Exécutées de manière itérative, pour chaque ligne
 - Coût non évalué (trou noir... *encore* !)
 - Pas de parallélisme : mono thread

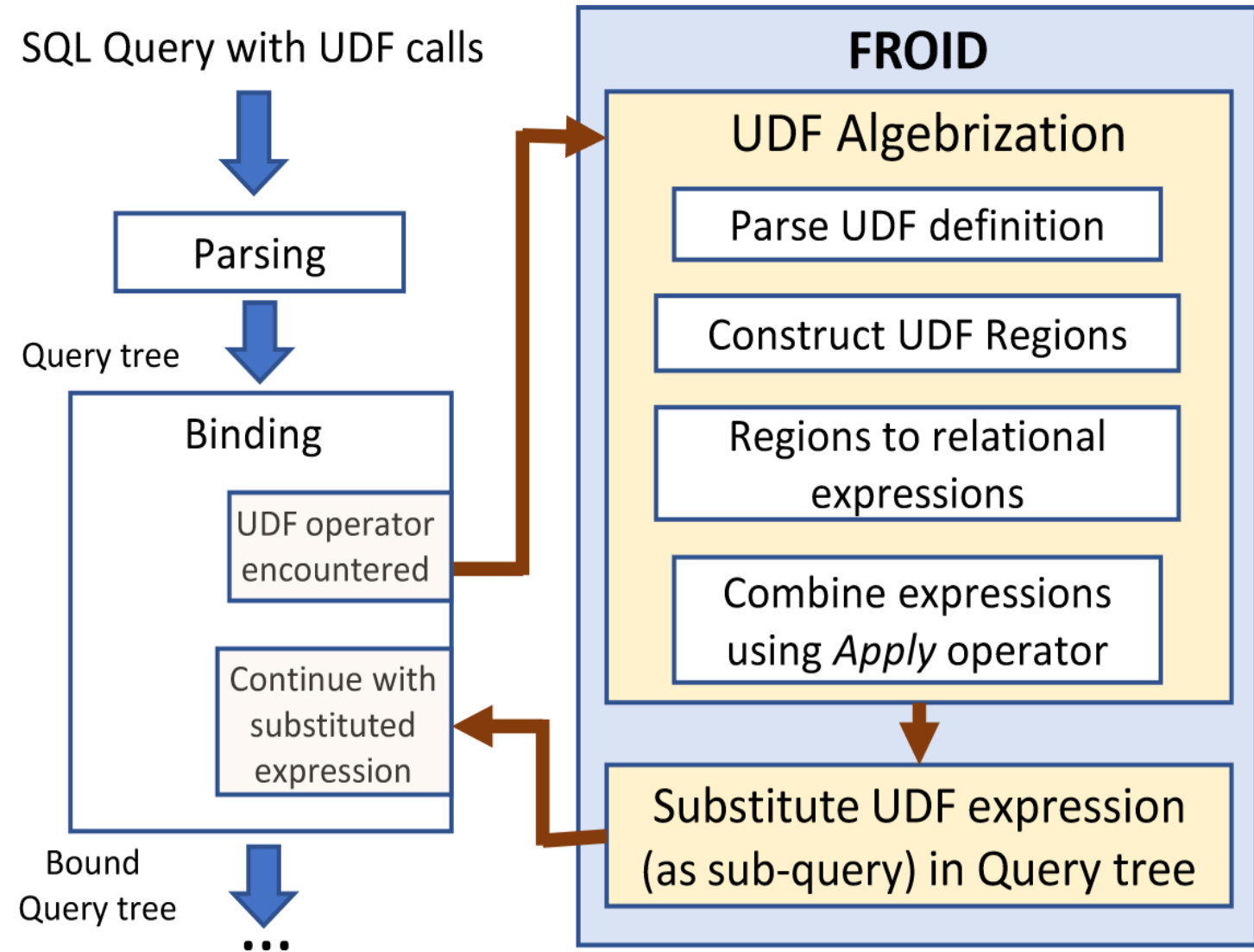


Scalar UDF inlining

- Nouvelle optimisation du moteur SQL 2019
- Transforme les UDF scalaires en expressions ou sous-requêtes dans le plan d'exécution
- Basé sur « *Froid* » (Microsoft Research), un framework pour l'optimisation des fonctions impératives dans les bases de données.
 - L'objectif de « *Froid* » est d'améliorer la programmation « impérative » et en l'occurrence dans SQL Server 2019 d'utiliser les UDF sans compromettre les performances

Fonctionnement

Source : Microsoft
*Froid: Optimization of
Imperative Programs in
a Relational Database*



Activation / désactivation

- Configuration nécessaire au niveau serveur :
 - Mode de compatibilité 150 (SQL 2019) : activé par défaut
- Peut être désactivé au niveau base de donnée
 - ALTER DATABASE SCOPED CONFIGURATION SET
TSQL_SCALAR_UDF_INLINING = ON|OFF;
- Peut être désactivé au niveau de la fonction
 - CREATE FUNCTION ... WITH INLINE ON|OFF;

Conditions

- Fonction (« inlineable ») transformée à l'exécution» si :
 - Se limite à : DECLARE, SET, SELECT, IF, ELSE, RETURN (pas de WHILE)
 - Ne doit pas faire appel à des fonctions non déterministes,
 - Ne doit pas référencer des variables ou paramètres de type table
 - Ne doit pas utiliser des types définis par l'utilisateur
 - N'est pas utilisée dans GROUP BY, DISTINCT, ORDER, CHECK ou une colonne calculée
 - Ne supporte pas EXECUTE AS (à part EXECUTE AS CALLER)
 - Pas de compilation en mode natif

Vérifier si la fonction est « inlineable »

- Colonne is_inlineable dans la DMV sys.sql_modules

```
SELECT OBJECT_NAME(object_id) name, is_inlineable
FROM sys.sql_modules
WHERE OBJECTPROPERTY(object_id, 'IsScalarFunction') = 1;
```
- L'optimiseur de SQL Server décide d'incorporer ou pas la fonction
 - Vérifier dans le plan de requête en XML si le nœud <UserDefinedFunction> est présent

Incorporation des fonctions scalaires : en bref

- Gain de performance potentiel pour certaines applications (abusant des fonctions scalaires)
 - Evite l'itération, meilleur calcul du coût, permet le parallélisme...
- Aucune modification de code, si le code est supporté (sinon le mode de fonctionnement ne change pas)
- Fonctionne avec toutes les éditions (SQL 2019)

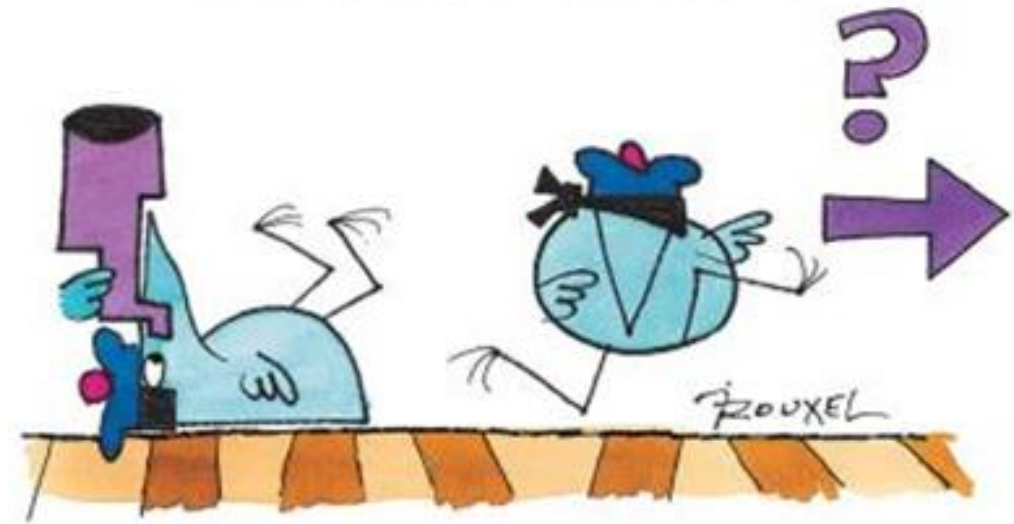
démo

4ème piste : utiliser le mode batch

- Mode batch sur columnstore
 - Depuis SQL Server 2012, uniquement avec un index columnstore
- Mode batch sur row store (nouveau 2019)
 - Le mode batch sur du *row store* peut permettre d'améliorer des requêtes massives même en l'absence d'index columnstore.
 - Réservé à l'édition Enterprise

La devise Shadok de la semaine

QUAND ON NE SAIT PAS OÙ L'ON VA,
IL FAUT Y ALLER...
... ET LE PLUS VITE POSSIBLE.



Row mode vs batch mode

- Row mode : l'unité de données traitées dans le plan d'exécution est la ligne
- Batch mode : l'unité de données traitées dans le plan est un ensemble d'au plus 900 lignes (batch)
 - Dédié aux requêtes massives (millions++ de lignes)
 - Moins d'instructions CPU par ligne traitée, optimisé pour le parallélisme, meilleure utilisation du cache CPU, meilleur débit de la mémoire
 - Gain de performance sensible

démo

5ème piste : poser automatiquement des index

- Indexation automatique (Azure)
 - Crée et/ou supprime automatiquement des index
 - A activer dans le portail Azure, niveau serveur et base

Hériter de : ⓘ

Serveur Valeurs Azure par défaut Ne pas hériter

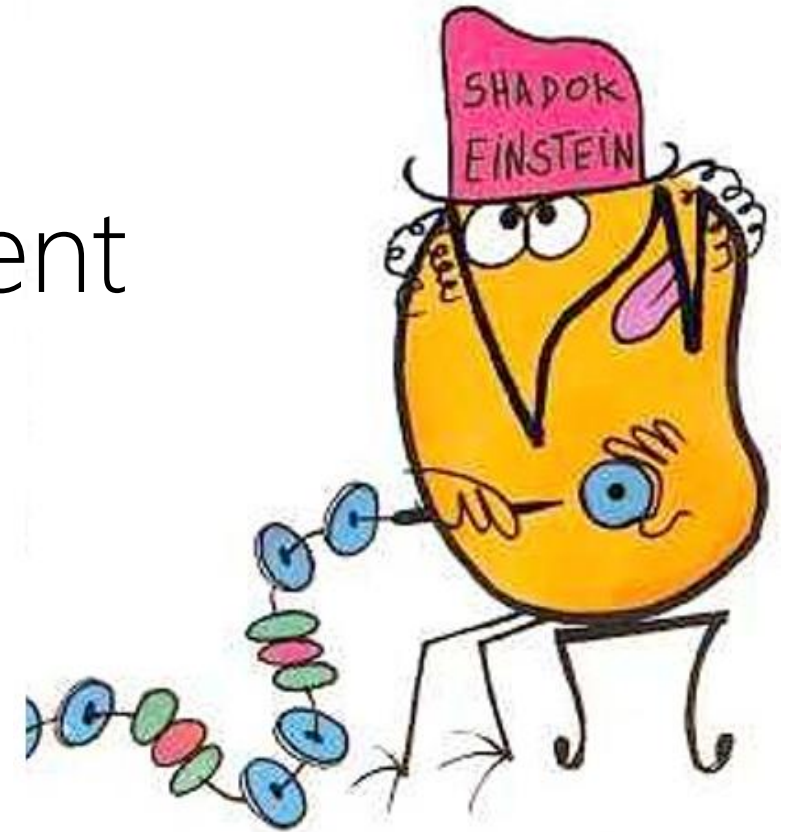
ⓘ La base de données hérite de la configuration de réglage automatique du serveur. Pour définir la configuration à hériter, accédez à : [Paramètres de configuration du serveur](#)

Configurer les options de réglage automatique ⓘ

Option	État souhaité	État actuel
 FORCER LE PLAN	ACTIVÉ DÉSACTIVÉ HÉRITER	ON Hérité du serveur
 CRÉER UN INDEX	ACTIVÉ DÉSACTIVÉ HÉRITER	OFF Hérité du serveur
 SUPPRIMER L'INDEX	ACTIVÉ DÉSACTIVÉ HÉRITER	OFF Hérité du serveur

Conclusion

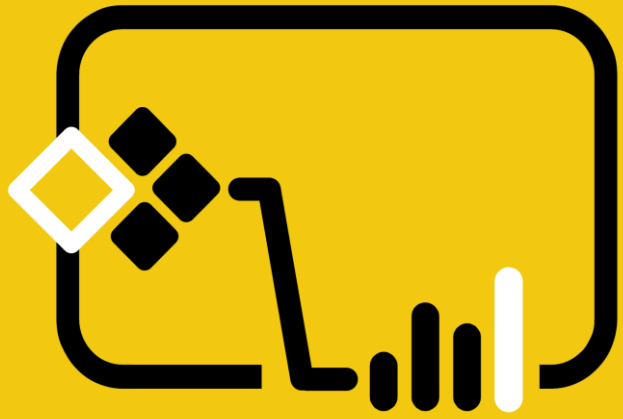
- L'optimiseur est-il plus intelligent qu'avant en votre absence ?
 - Edition Enterprise :
 - Jointures adaptatives (2017)
 - Feedback d'allocation mémoire (2017 + 2019)
 - Mode batch sur row store (2019)
 - Correction de plan automatique (2017)
 - Toutes éditions :
 - Execution entrelacée pour fonctions tables (2017)
 - Compilation différée de variable table (2019)
 - Incorporation des fonctions scalaires (2019)



IL VAUT MIEUX MOBILISER
SON INTELLIGENCE SUR DES
BETISES QUE MOBILISER
SA BETISE SUR DES CHOSES
INTELLIGENTES.

Questions





Merci!

Prenez 2 minutes pour
évaluer cette session

(RDV dans l'espace de conversation)