PROJECTION

RESTRICTION

MAGASIN	PRODUIT	PRIX	DATE	SITE
Auchan	Iphone	700	12/9/2017	Paris Est
Auchan	Iphone	760	12/9/2017	Paris Est
Leclerc	IPad	480	27/9/2017	Essonne
Fnac	IPad	870	30/9/2017	La Défense
Carrefour	PC Dell	600	24/9/2017	Paris Ouest
Leclerc	Galaxy	400	28/9/2017	Essonne
Boulanger	Galaxy	310	4/10/2017	Plaisir
Carrefour	Galaxy	470	4/11/2017	St. Quentin
Leclerc	MacBookAir	970	7/10/2017	Yvelines
Auchan	MacBookAir	1100	7/10/2017	Yvelines
Darty	PC Dell	860	19/10/2017	Velizy
Darty	PC Dell	640	23/10/2017	Parly
Darty	iPad	480	27/10/2017	Parly

MAGASIN	PRODUIT	PRIX	DATE	SITE
Auchan	Iphone	700	12/9/2017	Paris Est
Auchan	Iphone	760	23/9/2017	Paris Est
Carrefour	PC Dell	600	24/9/2017	Paris Ouest
Leclerc	IPad	400	27/9/2017	Essonne
Leclerc	Galaxy	400	28/9/2017	Essonne
Fnac	IPad	870	30/9/2017	La Défense
Fnac	iMac	1220	1/10/2017	La Défense
Fnac	iMac	2100	3/10/2017	Montparnasse
Boulanger	Galaxy	310	4/10/2017	Plaisir
...	...	...	...	...
...	...	...	...	...

PRODUIT	MARQUE	BREVET
Iphone	Apple	AP2056XY
PC Dell	Dell	DL3467GH
Galaxy	Samsung	SM734DG
IPad	Apple	AP205VY

JOINTURE

MAGASIN	PRODUIT	PRIX	DATE	SITE	MARQUE	BREVET
Auchan	Iphone	700	12/9/2017	Paris Est	Apple	AP2056XY
Auchan	Iphone	760	23/9/2017	Paris Est	Apple	AP2056XY
Carrefour	PC Dell	600	24/9/2017	Paris Ouest	Dell	DL3467GH
Leclerc	IPad	400	27/9/2017	Essonne	Apple	AP205VY
Leclerc	Galaxy	400	28/9/2017	Essonne	Samsung	SM734DG
Fnac	IPad	870	30/9/2017	La Défense	Apple	AP205VY
Boulanger	Galaxy	310	4/10/2017	Plaisir	Samsung	SM734DG
...	...	...	...	...		
...	...	...	...	...		

L'algèbre relationnelle offre cinq opérateurs de base grâce auxquels on peut exprimer un grand nombre de requêtes sur des données. À partir d'une table relationnelle *(le tableau bleu)*, la projection réduit le nombre de colonnes, la restriction celui des lignes, l'union fusionne deux tables de même structure, l'intersection calcule les lignes communes à deux tables de même structure, la jointure apparie deux tables sur un ou plusieurs critères.

> d'exploration de données, les tables seront rangées par lignes alors que pour les requêtes d'analyse agrégeant les valeurs, la représentation par colonne est plus adaptée.

Dans le même élan de définition de ce modèle élégant, des extensions ont été apportées pour traiter des objets plus complexes que les tables relationnelles, notamment les tables imbriquées et les tables à plus de deux dimensions. L'algèbre relationnelle a également été étendue à d'autres domaines spécialisés grâce à l'introduction, par exemple, d'opérations géométriques ou d'autres sur des données temporelles. Le champ d'investigation du modèle relationnel est loin d'être couvert !

DES ARBRES ET DES GRAPHES

Avec le Web, les bases de données ont évolué pour intégrer des données XML dont la structure est formalisée par des arbres. Une algèbre est aussi associée à ces arbres pour effectuer des recherches d'éléments et pour transformer leurs structures. La sélection et la projection permettent d'extraire des sous-arbres vérifiant certaines conditions portant soit sur les données (les valeurs d'un élément), soit sur les métadonnées (les éléments descriptifs de ces valeurs). Il est possible également de restructurer un arbre, ou de l'aplatir, pour obtenir une organisation différente des données et les visualiser sous des angles variés.

Enfin, l'appariement permet de comparer deux structures, une opération très utile pour de nombreuses applications, comme la bioinformatique, l'analyse d'images, les bases de documents... On associe à ce type d'opération une mesure de similarité qui peut être le nombre de transformations à effectuer pour obtenir un arbre à partir d'un autre. Plus la distance est petite, plus les arbres sont similaires. Cette notion est cruciale dans le contexte des *big data*. En effet, le calcul de ces distances est un problème difficile, dont la complexité augmente fortement avec le volume des données et avec la diversité structurelle des graphes à comparer. Les données dont nous avons parlé sont durables, mais qu'en est-il des flux de données, qui disparaissent juste après leur observation ?

LES FLUX DE DONNÉES

La logique de gestion est ici bien différente. Cette fois, ce sont les requêtes qui sont persistantes. Elles sont définies une fois pour toutes et agissent comme des filtres, déterminant les données observées qu'on souhaite conserver. Le premier opérateur indispensable sur un flux définit la taille de la fenêtre d'observation et un pas de glissement pour suivre le flux. Ces paramètres peuvent être définis par des durées, par le nombre d'événements attendus ou la combinaison des deux. Les données observées peuvent être stockées temporairement dans des tables relationnelles et filtrées par des requêtes définies en amont.

Pour faciliter l'interprétation des flux, il est souvent nécessaire de les corréler avec des données de référence persistantes dans des catalogues, ce qui impose des opérations de jointures potentiellement coûteuses si ces catalogues sont gigantesques. Par exemple, l'observation d'une température de −110 °C dans un bâtiment est sans doute liée à la défaillance d'un capteur. En revanche, une forte température trahit sans

doute l'explosion d'une chaudière située dans ce bâtiment. Cette interprétation n'est possible qu'avec un catalogue des immeubles avec leurs caractéristiques. Avec des millions de capteurs, le passage à l'échelle du traitement des flux envoyés est un défi. Le déploiement des compteurs Linky est un exemple.

LES LIMITES DES ALGÈBRES

Malgré tous ses avantages, les modèles de calcul «à la relationnelle» ne couvrent pas les besoins de l'ensemble des applications, et notamment ceux de deux grands domaines. Dans le premier, on trouve des applications dont les données sont conceptuellement relationnelles, mais dont les performances et certaines contraintes (telle la représentation en ligne des tables) ne sont pas satisfaisantes. C'est le cas des applications nécessitant une représentation des tables par colonnes plus adaptée aux calculs d'agrégats, ou dont la taille des tables, gigantesques, conduit à des performances médiocres de la part des systèmes de gestion de base de données.

Ce domaine est illustré par les applications décisionnelles faisant de gros calculs sur les colonnes et par des applications en astrophysique faisant des calculs sur des tables immenses. Dans les deux cas, même si conceptuellement on manipule des tables, on développe des algorithmes *ad hoc* opérant sur des données massivement distribuées et exploitant des architectures parallèles et des *clusters* de machines.

MapReduce permet une parallélisation des tâches en distribuant les données plutôt que le code

Le second domaine d'applications est illustré, au-delà des volumes de données importants, soit par des données non relationnelles (documents, images…), donc sans connaissance préalable d'une structure, soit par des logiques de traitement qui ne s'expriment pas aisément avec l'algèbre relationnelle. C'est le cas des algorithmes de classification ou d'apprentissage. Là encore, on a recours à une programmation *ad hoc*.

Notons que dans les deux domaines d'applications, le relationnel n'est pas exclu, mais seulement limité à quelques tâches. Par exemple, dans un projet de *machine learning*, la phase de prétraitement peut s'appuyer sur une technologie relationnelle, alors que la phase d'apprentissage proprement dite sera faite par un algorithme approprié.

Cette approche, notée NoSQL, n'est pas nouvelle, des interfaces entre les langages de requêtes et les langages de programmation ont toujours existé pour développer des applications complexes, sous-traitant aux systèmes de gestion de base de données ce qu'ils peuvent faire de mieux et laissant le programmeur réaliser les autres tâches. On la retrouve également dans les systèmes à base de *workflows* où l'on coordonne un ensemble d'activités pouvant être réalisées par des technologies différentes.

Aujourd'hui, ce mode d'ingénierie est toujours pratiqué, avec un regard plus spécifique sur ces tâches. Il s'agit en général d'opérations complexes dont l'optimisation nécessite la mobilisation de ressources de calcul particulières. Ce type de développement est généralement guidé par des patrons (ou *patterns*) de programmation qui intègrent l'invocation de services de données existant et le code de l'utilisateur.

Un de ces *patterns*, connu depuis longtemps en programmation fonctionnelle, est MapReduce (*voir la figure page suivante*). Il permet une parallélisation des tâches à grande échelle en distribuant les données au lieu de distribuer le code. Pour illustrer ce *pattern*, imaginons une opération visant à classifier un sac de billes en fonction de leurs couleurs et à évaluer ensuite la taille de chaque classe et le nombre total de billes. Cette opération peut être réalisée par une seule personne, mais elle sera longue, surtout si le sac de billes est très grand. On recrute alors dix enfants entre qui on partage les billes. En augmentant le parallélisme, on accélère le processus, mais on a à la fin dix classements, chacun produisant des paquets de billes selon leurs couleurs.

Après un travail intermédiaire de regroupement des paquets par couleur, on recrute un nouveau groupe d'enfants et on leur demande de compter le nombre de billes de chaque couleur. À la fin, une seule personne fait la somme des résultats intermédiaires.

Dans ce principe général du modèle de calcul MapReduce, la phase *map* répartit les données (les billes) sur plusieurs serveurs (les enfants), chacun réalisant la même tâche (reconnaissance de la couleur et classement des billes). La phase *reduce* prend les paquets de données intermédiaires et fait la tâche suivante (le comptage). Entre les deux, une «boîte noire» (le *shuffle*) regroupe les paquets par couleur, avant de les acheminer aux serveurs >