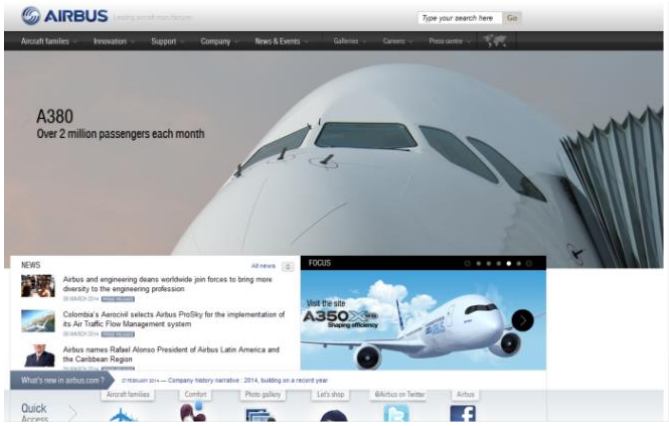
Un enseignement au numérique pour notre académie : les bases de données

Un enrichissement exponentiel de la toile

Marie-Danièle CAMPION
Recteur de l'académie de Clermont-Ferrand

Que ce soit au niveau des archives, des catalogues, des registres, des plans industriels qui étaient jusqu'alors transmis sous forme papier avec tous les problèmes de conservation et d'accès que cela pouvait engendrer, le numérique apporte aujourd'hui ses capacités de stockage et la rapidité de consultation des données conservées dans ce que l'on appelle des bases de données.



Le site de l'entreprise Airbus : véritable catalogue numérique

Ces bases de données sont un élément essentiel de l'activité sur Internet que ce soit de façon ouverte avec la mise en ligne par exemple de registres historiques, géographiques, économiques ou scientifiques ou de façon plus dissimulée avec la constitution de registres sur les centres d'intérêt des personnes et leurs modes de consommation.

Le site Mémoire des hommes donne accès à 1,3 million de fiches individuelles de militaires morts pour la France au cours de la Grande Guerre.

Dans la continuité des deux documents déjà produits par notre académie qui visent à apporter aux professeurs et aux élèves des éléments conceptuels sur le numérique, ce texte montre comment sont constituées physiquement les bases de données. Nous souhaitons ici fournir à nos élèves des éléments de réponse à la question de savoir comment l'on arrive à constituer ces bases informatiques de données et comment l'on procède pour rendre ensuite ces informations parfaitement accessibles. Il ne s'agit pas ici de savoir comment l'on fabrique une page web mais comment l'on arrive à concentrer puis à excentrer de l'information.

Le site du musée du Louvre permet une visite virtuelle de certaines salles

Nous assistons actuellement à ce que l'on peut qualifier d'enrichissement exponentiel de la toile : le nombre de données présentes sur internet croît en effet de façon exponentielle. C'est cet enrichissement gigantesque qui ouvre aujourd'hui le champ du **Big Data** qui correspond à l'ensemble des données exploitées ou non qui circulent sur Internet et qui seront au cœur des enjeux scientifiques, commerciaux et sociétaux du XXIème siècle. Nul doute que le Big Data fournira à nos élèves des champs professionnels d'envergure ; le métier de « data scientist » fait en l'occurrence partie des métiers du proche avenir qu'il nous appartient d'intégrer dans la palette de leurs possibilités d'orientation.

Les systèmes de bases de données relationnels[1]

Serge Abiteboul
Directeur de recherche à l'INRIA
Membre de l'Académie des sciences

> Je ne connais pas d'être vivant, de cellule, tissu, organe,
> individu et peut-être même espèce, dont on ne puisse pas dire
> qu'il stocke de l'information, qu'il traite de l'information,
> qu'il émet et qu'il reçoit de l'information.
> *Michel Serres*

L'information stockée, traitée, échangée, est au cœur de l'activité des êtres vivants, des objets du monde, des associations humaines. Les systèmes informatiques, en nous aidant à gérer de l'information, représentée sous forme numérique, ont transformé nos vies en profondeur. Pour obtenir de l'information, nous pouvons interroger un *système de gestion de bases de données*. Pour ce faire, nous nous exprimons dans un langage informatique simple, peut-être graphique, peut-être même dans notre langue naturelle[2]. Le système traduit cette demande dans un langage formel. Par cela, nous entendons une syntaxe qui permet au système de préciser la demande de l'utilisateur, et une sémantique formelle qui donne un sens exact à cette syntaxe. La logique mathématique offre un tel langage formel. Nous évoquerons ici les systèmes de gestion de bases de données et leurs liens profonds avec la *logique du premier ordre*.

Nous avons typiquement, d'un côté, un serveur de données quelque part sur le Web, avec des disques et leurs pistes qui gardent précieusement des séquences de bits, des structures d'accès compliquées comme des index ou des arbres-B, des hiérarchies de mémoires avec leurs caches et, de l'autre, un utilisateur. Supposons que le serveur soit celui d'IMDb, qui gère une base de données sur le cinéma. Supposons que l'utilisateur, disons Alice, veuille savoir quels films ont été réalisés par Alfred Hitchcock. Pour ce faire, elle spécifie des mots-clés ou remplit les champs d'un formulaire proposé par IMDb. Sa question voyage depuis son navigateur

[1] Tiré de la leçon inaugurale du cours au Collège de France : http://www.college-de-france.fr/site/serge-abiteboul/inaugural-lecture-2012-03-08-18h00.htm.

[2] Nous entendons par langues « naturelles » des langues élaborées dans le temps par des groupes de locuteurs, comme le français ou l'anglais. Ceci est moins par opposition avec des langues « construites » comme l'espéranto, qu'avec des langages formels comme la logique du premier ordre, SQL ou Java.

jusqu'au serveur de données. Là, cette question est transformée en un programme peut-être complexe qui s'exécute pour obtenir la réponse. Ce qui est important : ce programme, Alice n'a pas envie de l'écrire ; d'ailleurs, elle n'a pas à l'écrire.

Le système élémentaire qui permet de gérer des données est un *système de fichiers*. Un fichier est une séquence de bits qui peut représenter une chanson, une photo, une vidéo, un courriel, une lettre, un roman, etc. Votre ordinateur personnel et votre téléphone stockent leurs données dans des systèmes de fichiers. Et parfois quand vous ne savez plus où vous avez mis quelque chose, vous faites des « recherches » dans ces système de fichiers. C'est rudimentaire. Nous parlons ici de systèmes qui gèrent aussi des données mais qui sont bien plus sophistiqués que les systèmes de fichiers, les *systèmes de gestion de bases de données*. Ce sont des logiciels complexes, résultats de dizaines d'années de recherche et de développement. Ils permettent à des individus ou à des programmes d'exprimer des *requêtes* pour interroger des bases de données ou pour les modifier. Nous nous focaliserons ici sur les plus répandus de ces systèmes, les *systèmes relationnels*, parmi lesquels nous trouvons des logiciels commerciaux très répandus comme celui d'Oracle et des logiciels gratuits très utilisés comme MySQL.

Le calcul et l'algèbre relationnels

Un système de gestion de bases de données sert de médiateur entre des individus et des machines. Pour mieux s'adapter aux individus, il doit organiser et présenter les données de façon intuitive. Il doit aussi proposer un langage, pour exprimer des requêtes, facilement utilisable par des êtres humains. Ces exigences forment le point de départ du modèle relationnel[3] proposé par Ted Codd, un chercheur d'IBM, dans les années 1970. Des mathématiciens avaient développé à la fin du XIX^{e} siècle (bien avant l'invention de l'informatique et des bases de données) la *logique du premier ordre*, pour formaliser le langage des mathématiques. Codd a eu l'idée d'adapter cette logique pour définir un modèle de gestion de données, le *modèle relationnel*.

Figure 1. Une base de données relationnelles

Film

Titre	Réalisateur	Acteur
Casablanca	M. Curtiz	Humphrey Bogart
Casablanca	M. Curtiz	Peter Lorre
Les 400 coups	F. Truffaut	Jean-Pierre Léaud
Star Wars	G. Lucas	Harrison Ford

Séance

Titre	Salle	Heure
Casablanca	Lucernaire	19:00
Casablanca	Studio	20:00
Star Wars	Sel	20:30
Stars Wars	Sel	22:15

[3] Serge Abiteboul, Richard Hull et Victor Vianu, *Foundations of databases*, Addison-Wesley, 1995 : www.webdam.inria.fr/Alice.

Dans le modèle relationnel, les données sont organisées en tableaux à deux dimensions que nous appellerons des *relations*. À la différence des mathématiciens, nous supposons les relations de taille finie. Comme illustration, nous utiliserons une base de données consistant en une relation *Film* et une relation *Séance* (Figure 1**.**). Une ligne de ces relations est appelée un *n-uplet* où *n* est le nombre de colonnes. Par exemple, ⟨Star Wars, Sel, 22:15⟩ est un 3-uplet, un triplet, dans la relation *Séance.* Les colonnes ont des noms, appelés *attributs,* comme *Titre.*

Les données sont interrogées en utilisant comme langage le *calcul relationnel*. Le calcul relationnel (très fortement inspiré de la logique du premier ordre) s'appuie sur des noms qui représentent des *relations* comme *Film* ou *Séance*, des *entrées* de ces relations comme « Star Wars », des *variables* comme *t*, *h*, et des *symboles logiques*, $\wedge$ (et), $\vee$ (ou), $\neg$ (non), $\Rightarrow$ (implique), $\exists$ (existe), $\forall$ (pour tout). Avec tout ça, des formules logiques peuvent être construites comme :

$$q_{HB} = \exists\, t, r\ (\ \textit{Film}(t, r, \text{« Humphrey Bogart »}) \wedge \textit{Séance}(t, s, h)\ )$$

Si cela vous paraît cryptique, en français, cela se lit : il existe un titre *t* et un réalisateur *r* tels que le n-uplet ⟨ *t*, *r*, « Humphrey Bogart » ⟩ se trouve dans la relation *Film*, et le n-uplet ⟨ *t*, *s*, *h* ⟩ dans *Séance*. Observez que *s* et *h* ne sont pas quantifiées dans la formule précédente ; nous dirons que ces deux variables sont *libres*. La formule q_{HB} peut être vue comme une *requête du calcul relationnel.* Elle se lit alors : donnez-moi les salles *s* et les horaires *h*, s'il existe un réalisateur *r* et un titre *t* tels que... En d'autres termes, « Où et à quelle heure puis-je voir un film avec Humphrey Bogart ? ». Ce langage, le calcul relationnel, permet d'exprimer des questions dans une syntaxe qui évite les ambiguïtés de nos langues naturelles. Si elles pouvaient aimer, les machines aimeraient la simplicité, la précision du calcul relationnel. En pratique, elles utilisent le langage SQL (Structured Query Language) qui exprime différemment les mêmes questions. Par exemple, la question précédente s'exprime en SQL comme :

select salle, heure

from Film, Séance

where Film.titre= Séance.titre **and** acteur= « Humphrey Bogart »

C'est presque compréhensible. Non ? Et qu'Alice s'exprime en français ou qu'elle utilise une interface graphique, le système transforme sa question en requête SQL[4].

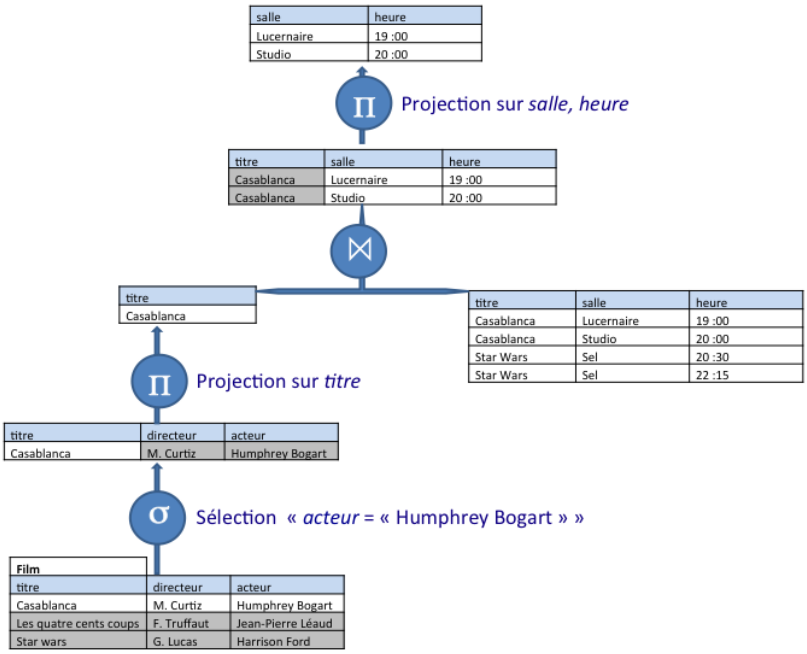
La question du calcul relationnel précédente (ou en SQL) précise bien ce qu'Alice demande. Cette question a un sens précis, une sémantique. Elle définit une réponse, un ensemble de n-uplets. Ce que la question ne dit pas c'est comment calculer la réponse. Pour le « comment », on utilise l'*algèbre relationnelle* introduite par Codd*.* Une étape importante consiste à transformer une question du calcul en une expression algébrique qui permet de calculer la réponse à cette question.

[4] SQL va plus loin que le calcul relationnel. Par exemple, il permet d'ordonner les résultats et d'appliquer des fonctions simples comme la somme ou la moyenne.

L'algèbre relationnelle consiste en un petit nombre d'opérations de base qui, appliquées à des relations, produisent de nouvelles relations. Ces opérations peuvent être composées pour construire des expressions algébriques de plus en plus complexes. Pour répondre à la question qui nous sert d'exemple, il nous faudra trois opérations, la *jointure*, la *sélection* et la *projection*, que nous composerons dans l'expression suivante de l'algèbre relationnelle :

$$E_{HB} = \Pi_{salle,heure} (\Pi_{titre} (\sigma_{acteur = « Humphrey Bogart »}(Film)) \bowtie Salle)$$

Figure 2. L'évaluation d'une requête algébrique



Nous pourrons suivre l'évaluation de cette expression algébrique en figure 2. L'opération de sélection, dénotée σ, filtre une relation, ne gardant que les n-uplets satisfaisant une condition, ici *acteur = « Humphrey Bogart »*. L'opération de projection, dénotée Π, permet aussi de filtrer de l'information d'une relation mais cette fois en éliminant des colonnes. L'opération peut-être la plus exotique de l'algèbre, la « jointure », dénotée $\bowtie$, combine des n-uplets de deux relations. D'autres opérations non illustrées ici permettent de faire l'union et la différence entre deux relations ou de renommer des attributs. La puissance de l'algèbre relationnelle tient de la possibilité de composer ces opérations. C'est ce que nous avons fait dans l'expression algébrique E_{HB} qui permet d'évaluer la réponse à la question q_{HB}.

Notre présentation est rapide mais il est important que le lecteur comprenne l'intérêt de l'algèbre. Il est relativement simple d'écrire un programme qui évalue la réponse à une question du calcul relationnel. Il est plus délicat d'obtenir un programme qui calcule cette réponse *efficacement*. L'algèbre relationnelle découpe le travail. Un programme particulier très efficace peut être utilisé pour chacune des opérations de l'algèbre ; le résultat est obtenu en composant ces programmes. L'efficacité provient notamment de ce que les opérations considèrent des ensembles de n-uplets plutôt que les n-uplets un à un.

Nous pouvons donc exprimer une question en calcul relationnel, et le système peut traduire cette question en expression algébrique et calculer efficacement sa réponse. Pourtant, quand Codd proposa cette approche, la réaction des ingénieurs qui géraient alors de gros volumes de données et de grandes applications, fut unanime : « trop lent ! Ça ne passera pas à l'échelle ». Ils se trompaient. Pour traduire l'idée de Codd en une industrie de milliards de dollars, il manquait l'optimisation de requête. Après des années d'effort, les chercheurs sont parvenus à faire fonctionner les systèmes relationnels avec des temps de réponse acceptables. Avec ces systèmes, le développement d'applications gérant des données devenait beaucoup plus simple ; cela se traduisait par un accroissement considérable de la productivité des programmeurs d'applications gérant des gros volumes de données.

Pour conclure, nous pouvons mentionner de nombreuses autres tâches que les systèmes relationnels accomplissent à côté de l'évaluation de requêtes : la gestion des transactions, des pannes, les droits d'accès de chaque utilisateur, l'archivage, etc.