TECHNIQUES
DE L'INGÉNIEUR

Réf. : **H3850 V1**

Date de publication :
10 décembre 1993

Bases de données réparties

Cet article est issu de : **Archives**

par **Claude CHRISMENT, Geneviève PUJOLLE, Gilles ZURFLUH**

Pour toute question :
Service Relation clientèle
Techniques de l'Ingénieur
Immeuble Pleyad 1
39, boulevard Ornano
93288 Saint-Denis Cedex

Par mail :
infos.clients@teching.com
Par téléphone :
00 33 (0)1 53 35 20 20

Bases de données réparties

par **Claude CHRISMENT**
Docteur ès Sciences
Professeur d'informatique à l'Université Toulouse III

Geneviève PUJOLLE
Maître de Conférences en informatique à l'Université Toulouse I

et **Gilles ZURFLUH**
Docteur ès Sciences
Professeur d'informatique à l'Université Toulouse I

Alors que sont connues depuis plusieurs années les techniques de répartition des bases de données, peu d'entreprises ont adopté jusqu'à présent ce type de systèmes informatiques. La plupart des ingrédients nécessaires à la mise en place de ces systèmes est pourtant présente dans les organisations : réseaux locaux et publics, micro-ordinateurs puissants et stations de travail, systèmes de bases de données relationnelles sur gros et micro-ordinateurs.

Cet article présente les caractéristiques essentielles des systèmes de gestion de bases de données réparties. Répartir des données sur des ordinateurs indépendants mais interconnectés par un réseau suppose la mise en œuvre de mécanismes chargés de décrire les données, décomposer les requêtes d'accès aux données, assurer l'intégrité globale de la base répartie.

Dans le paragraphe 1, un rapide historique de l'évolution des systèmes informatiques montre que les entreprises réunissent actuellement les conditions nécessaires à la mise en place de bases de données réparties. Le paragraphe 2 présente une définition d'une base de données répartie ainsi que l'architecture et les fonctions des systèmes chargés de les gérer. Nous présentons ensuite successivement les principes de conception, les mécanismes de traitement des requêtes, les principes de localisation des données ainsi que l'optimisation et la validation des traitements répartis.

1. Évolution des bases de données

Avant de donner une définition technique des bases de données réparties, nous allons préciser la place qu'occupent les bases de données dans les systèmes d'information d'une entreprise.

Le système d'information correspond à un ensemble de moyens et de processus assurant le traitement de l'information à des fins de décisions. Plus précisément, un système d'information est constitué :

— d'une base de données (au sens large, c'est-à-dire fichiers classiques ou base de données), véritable mémoire de l'organisation puisqu'elle contient des informations relatives au personnel, aux produits, aux clients de l'entreprise ;

— d'un ensemble de processus (et donc de processeurs humains ou automatiques) capables de traiter ces données.

Le rôle essentiel d'un système d'information est de permettre le « pilotage » de l'entreprise en fournissant aux décideurs (dirigeants, gestionnaires,...) des données fiables et synthétiques sur l'état de l'entreprise et de son environnement.

La rapidité et l'efficacité d'une prise de décision sont naturellement induites par la qualité du système d'information. Cette qualité est notamment liée à la cohérence de la base de données qui doit refléter le plus fidèlement possible la situation de l'entreprise. Par exemple, la quantité d'un produit enregistrée dans un fichier doit correspondre à la quantité réellement disponible en magasin.

1.1 Systèmes centralisés

Pour obtenir la cohérence des données stockées, les informaticiens ont naturellement songé à centraliser la base de données (ou les fichiers). Ainsi, les données sont stockées sur un processeur unique auquel les utilisateurs accèdent à partir de terminaux.

Cette solution a l'avantage de la simplicité ; les informaticiens ont l'entière maîtrise de l'informatique : choix cohérent des matériels et logiciels, application d'une méthode et de standards dans le développement et la maintenance des programmes, contrôle de la cohérence des données grâce à une administration centralisée des données. D'autre part, cette solution correspond bien à la technologie des années 60 et 70 où un système informatique est généralement constitué d'un ordinateur central auquel sont connectés un ensemble de terminaux passifs.

1.2 Systèmes décentralisés

Face à la multiplication des micro-ordinateurs et des stations de travail dans les entreprises au début des années 80, les informaticiens ont été conduits (bon gré, mal gré) à décentraliser le traitement de l'information. Par exemple, une des formes de décentralisation appelée **Infocentre**, consiste à remplacer les terminaux des utilisateurs par des micro-ordinateurs connectés au processeur central ; le principe de fonctionnement de l'Infocentre est alors le suivant :

— le stockage et les mises à jour de la base de données sont réalisés sur le processeur central ;

— les traitements autres que les mises à jour peuvent être conçus, réalisés et mis en œuvre par les utilisateurs directement sur leurs micro-ordinateurs.

Les avantages d'un tel système sont indéniables :

— la cohérence de la base de données est préservée grâce à la centralisation du stockage et des mises à jour ;

— l'utilisateur peut développer localement ses propres applications sur son micro-ordinateur à partir de données extraites de la base centrale : utilisation de tableurs, de logiciels de statistiques, d'éditeurs d'états, de L4G (Langages de 4e Génération) etc. ;

— on évite le développement anarchique d'applications gérant des données stockées localement sur des micro-ordinateurs ;

— les utilisateurs peuvent concevoir et réaliser des applications simples (à partir de L4G) déchargeant ainsi les informaticiens de ce type de développement.

1.3 Systèmes répartis

Dans les systèmes centralisés et décentralisés, le fonctionnement du système d'information automatisé repose en grande partie sur la disponibilité du processeur central qui gère la base de données.

Dans un système réparti, un ensemble de processeurs autonomes reliés par un réseau de communication coopèrent pour assurer la gestion des informations.

Un tel système présente des avantages certains. Tout d'abord, en préservant certains apports de la solution décentralisée, le système réparti permet de limiter la vulnérabilité des centres informatiques. Le principe est simple : le système d'information (données et traitements) est réparti sur différents sites (processeurs de traitement autonomes) interconnectés par un réseau de communication. Ainsi, la défaillance d'un site ne peut entraîner à elle seule l'indisponibilité totale du système.

D'autre part, ce type de système préserve l'autonomie des sites en permettant à un groupe d'utilisateurs de créer et de gérer sa propre base de données tout en autorisant un accès aux autres utilisateurs *via* le réseau.

Véritable système multiprocesseurs, un système réparti peut sensiblement améliorer les performances des traitements : en effet, avec une localisation des données et une répartition des traitements bien étudiées, la déperdition induite par les communications des données intersites peut être compensée par le gain issu du parallélisme dans l'exécution des traitements (il convient néanmoins d'éviter de comparer les performances brutes des systèmes répartis et des systèmes centralisés sans prendre en compte d'autres critères d'évaluation).

Un système réparti localement ou géographiquement peut s'adapter aisément à la structure d'une entreprise (services, départements, succursales...) et refléter ainsi une organisation particulière.

Un autre avantage du système réparti est sa modularité qui permet d'accroître aisément sa taille par adjonction de nouveaux sites.

Mais les systèmes répartis présentent aussi des inconvénients, notamment :

— la complexité des mécanismes de décomposition de requêtes, de synchronisation des traitements, de maintien de la cohérence ;
— les coûts induits par les transmissions de données sur les réseaux locaux et surtout publics ;
— les compétences requises des informaticiens pour concevoir les logiciels applicatifs.

1.4 Architecture des systèmes répartis

L'appellation système réparti recouvre diverses architectures depuis les architectures client-serveur jusqu'aux architectures totalement réparties.

L'architecture client-serveur considère deux types de processeurs généralement distincts :

— les serveurs qui offrent un service à des clients, par exemple un serveur base de données ou un serveur imprimante ;
— les clients qui émettent des requêtes aux serveurs pour les besoins d'une application.

Dans l'architecture client-serveur la plus simple, la quasi-totalité du système de gestion de base de données (SGBD) se trouve sur le serveur, les processeurs clients ne disposant que des mécanismes de décodage et de transmission des requêtes vers ce serveur.

Une architecture totalement répartie est une généralisation de l'architecture client-serveur : les processeurs sont autonomes dans le sens où ils peuvent disposer d'un SGBD et assurer la pleine gestion d'une base de données locale. En plus, s'ils ne disposent pas des ressources nécessaires à une application qui leur est soumise, ils déterminent la localisation des données et des traitements qui leur sont nécessaires et établissent une coopération avec les processeurs détenteurs de ces ressources. Cette architecture permet d'éviter la présence du goulet d'étranglement d'un serveur base de données puisque les données sont réparties voire dupliquées dans le réseau.

2. Bases de données réparties : principes et définition

L'évolution des techniques informatiques depuis les vingt dernières années permet d'adapter les outils informatiques à l'organisation des entreprises (et non l'inverse). La puissance des micro-ordinateurs et des stations de travail, la fiabilité et la souplesse des SGBD relationnels, les performances des réseaux permettent d'envisager une répartition des ressources informatiques tout en préservant l'essentiel, c'est-à-dire la cohérence des bases de données.

2.1 Définition

Une base de données répartie (BDR) est un ensemble structuré et cohérent de données, stocké sur des processeurs distincts, et géré par un système de gestion de bases de données réparties (SGBDR) (figure 1).

Cette définition mérite toutefois quelques précisions.

Le SGBDR repose sur un système informatique réparti qui est constitué d'un ensemble de processeurs autonomes appelés sites (mini ou micro-ordinateurs, stations de travail,...) reliés par un réseau de communication (local ou public) qui leur permet d'échanger des données. Un SGBDR suppose en plus que les données soient stockées sur deux processeurs au moins, ceux-ci étant dotés de leur SGBD propre. Ainsi, dans l'exemple d'une configuration constituée de trois processeurs interconnectés, dont l'un est chargé de la gestion des données (serveur base de données), la base de données n'est évidemment pas répartie bien que l'on soit en présence d'un système réparti.

La BDR est décrite par un schéma global qui contient la localisation des données et qui permet ainsi d'aiguiller un traitement sur les processeurs détenant les données à traiter.

Lorsque les BD qui composent la BDR reposent sur des modèles de données différents (hiérarchique, réseau, relationnel, orienté objet), on parle de **BDR hétérogène**. Par voie de conséquence, le SGBDR est également hétérogène puisqu'il est constitué de SGBD de types différents.

2.2 Architecture

Dans un SGBDR, il apparaît une dualité entre le **niveau global** qui ouvre l'accès à la BDR, et le **niveau local** où interviennent les différents SGBD qui manipulent directement les bases de données (figure 2).

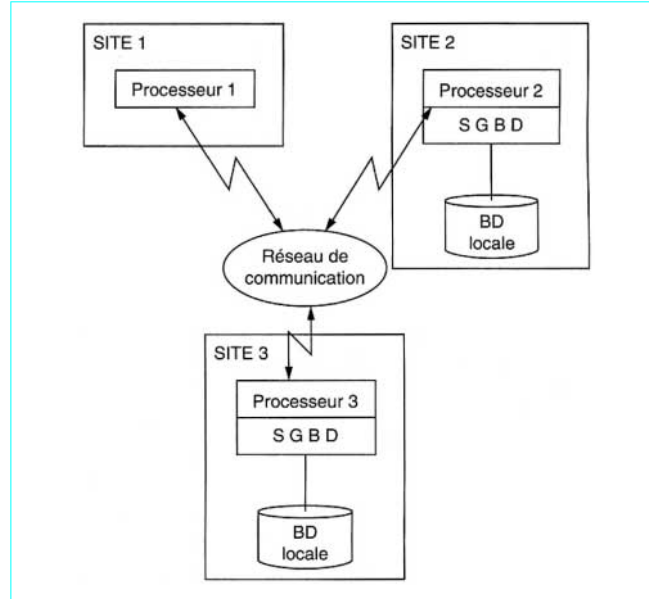


Figure 1 – Principe d'un SGBDR : système réparti + base de données répartie (ensemble des bases de données locales)

Si l'on affine l'architecture de la figure 2, on s'aperçoit que les couches fonctionnelles du niveau global (SGBDR) correspondent à celles du niveau local (SGBD), comme le montre la figure 3.

Nous allons expliciter chacune des fonctions qui apparaissent au niveau global.

2.3 Fonctions

Les fonctions suivantes mettent en lumière l'originalité d'un SGBDR par rapport à un SGBD. Nous considérons que le réseau de communication assure le transport et le contrôle des messages entre les différents processeurs du système réparti.

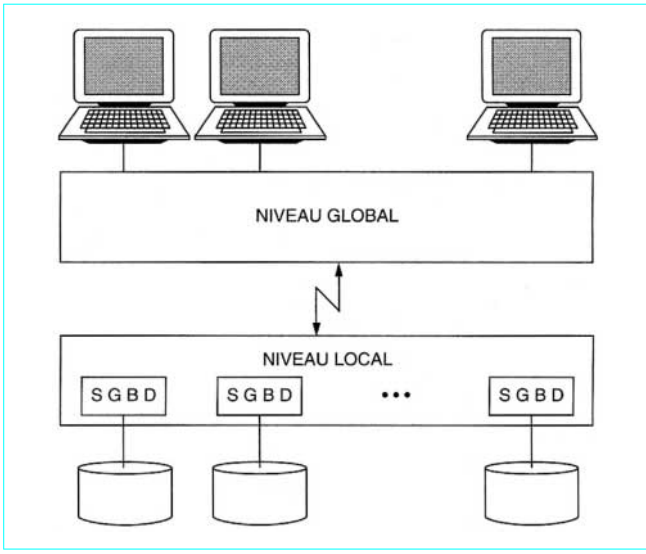


Figure 2 – Niveaux local et global d'un SGBDR

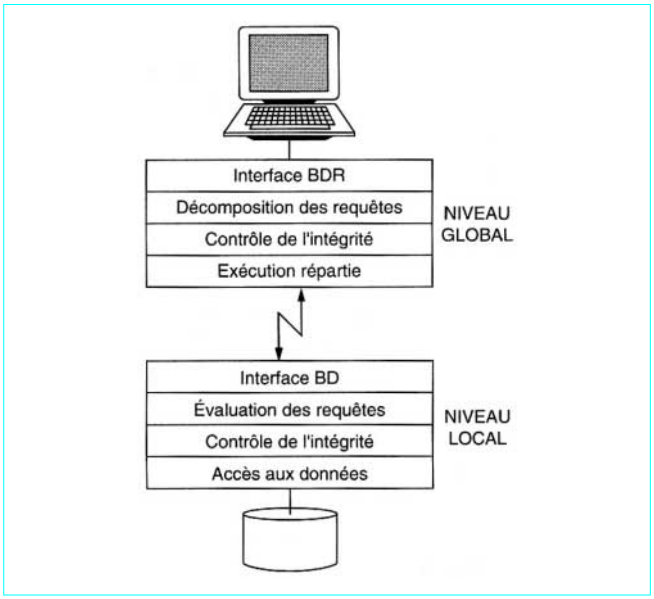


Figure 3 – Décomposition fonctionnelle d'un SGBDR

2.3.1 Interface d'une BDR

Comme toute base de données, une BDR est décrite dans un dictionnaire de données sous la forme de schémas globaux distincts conformément à l'architecture ANSI/SPARC :

— le **schéma conceptuel** où les données sont représentées sans prendre en compte des contraintes techniques ou de mise en forme ; toutes les données sont décrites dans ce schéma, indépendamment de leur localisation dans le système réparti ;

— les **schémas externes** où les données sont décrites sous forme de vues, chacune d'elles étant adaptée à une classe particulière d'utilisateurs ; un schéma externe, élaboré à partir du schéma conceptuel, peut naturellement mixer des données stockées dans différentes bases ;

— le **schéma interne** où sont notamment spécifiées la fragmentation des données et la localisation de ces fragments ; les données y sont décrites en fonction de l'architecture du système réparti et des spécificités techniques du système informatique (diversité des matériels et des modèles de données, architecture du réseau,...).

C'est au travers de ces différents schémas que l'utilisateur (programmeur ou utilisateur final) peut accéder aux données réparties ; le langage utilisé est généralement un langage de type SQL (article *Langages de bases de données* : *SQL et les évolutions vers l'objet* [H 3 128] dans ce traité).

2.3.2 Décomposition des requêtes

Un traitement réparti fait appel à des données gérées par des SGBD distincts (nous excluons de notre présentation les problèmes liés à la répartition des traitements). Un traitement réparti contient donc des requêtes formulées à partir d'un schéma externe global ; ces requêtes correspondent à un ensemble d'opérations de recherches et de mises à jour sur des données de la BDR. Le SGBDR contrôle et analyse chaque requête et la décompose en opérations locales (plan d'exécution réparti) qui seront soumises pour exécution aux SGBD concernés (cet aspect est développé dans le paragraphe 4).

2.3.3 Contrôle de l'intégrité

Le contrôle de l'intégrité des données par un système informatique permet d'assurer que tout traitement (transaction) sur la base de données fait passer celle-ci d'un état cohérent (ou supposé tel) à un autre état cohérent. Nombreuses sont les sources pouvant engendrer des anomalies : absence d'expression de certaines contraintes d'intégrité dans le schéma des données, perte d'opérations suite à un enchevêtrement de mises à jour concurrentes, panne du réseau, etc.

Ce type de problème existe aussi dans les SGBD centralisés qui proposent des solutions adaptées notamment pour gérer les accès concurrents aux données (techniques d'évitement ou de détection des incohérences).

3. Conception des bases de données réparties

L'existence de SGBDR, aussi sophistiqués soient-ils, ne dispense pas l'utilisateur (l'administrateur des données) de concevoir la BDR, c'est-à-dire de définir la structure de la base de données et les opérations qui lui sont applicables. Le problème de conception est cependant différent selon que l'on crée de toute pièce une BDR (démarche descendante) ou bien que l'on constitue une BDR par agrégation de bases de données existantes (démarche ascendante).

3.1 Démarche descendante

Aux niveaux conceptuel et externe, la BDR est perçue comme une base de données centralisée ; les processus de conception classiques pour bases de données centralisées s'appliquent donc aux niveaux externe global et conceptuel global.

Toute la difficulté réside donc dans le niveau interne global où, considérant la BDR comme un ensemble de relations, on spécifie :

— la fragmentation des relations en unités de localisation ;
— la localisation de ces fragments dans le réseau.

L'ensemble des fragments stockés sur un site donné correspond à une base de données locale. La figure 4 présente l'architecture d'une BDR. On constate la présence des trois niveaux de l'architecture ANSI/SPARC (externe, conceptuel et interne) à la fois au niveau global (celui de la BDR) et au niveau local (celui des BD).

3.1.1 Fragmentation

Grâce à sa simplicité, à l'universalité de ses concepts et surtout à l'algèbre qui lui est associée, le modèle relationnel se prête bien à l'étude de la répartition des données. Si bien que l'on a qualifié ce modèle de **pivot** en lui faisant jouer un rôle central : les bases de données conçues avec d'autres modèles (hiérarchique et réseau [1]) sont alors traduites en termes relationnels pour les intégrer dans le système réparti.

Pour fragmenter une relation globale sans perte d'information, il suffit d'appliquer à cette relation l'opération algébrique de restriction (fragmentation horizontale) ou celle de projection (fragmentation verticale). Les opérations de jointure et d'union permettent ensuite de reconstituer la relation initiale.

Considérons la relation ASSURES comportant un numéro d'assuré, un nom, une ville, un type et un montant de contrat pour des personnes habitant soit Toulouse, soit Paris.

ASSURES	NAS	NOM	VILLE	TYPECT	MT-CT
	1024661J	DEXTER	TOULOUSE	1	3224
	3015248K	BERNIE	PARIS	3	5632
	5040283A	PICCOLI	TOULOUSE	3	5845
	7320125C	DUPUY	PARIS	2	9872

Il est possible de spécifier des fragments (en utilisant un langage de type SQL pour spécifier des vues [1]) comme suit :

```
define fragment FR1
       as select  NAS, NOM, VILLE
          from    ASSURES
          where   VILLE = 'TOULOUSE'
define fragment FR2
       as select  NAS, TYPECT, MT-CT
          from    ASSURES
          where   VILLE = 'TOULOUSE'
define fragment FR3
       as select  NAS, NOM, VILLE
          from    ASSURES
          where   VILLE = 'PARIS'
define fragment FR4
       as select  NAS, TYPECT, MT-CT
          from    ASSURES
          where   VILLE = 'PARIS'
```

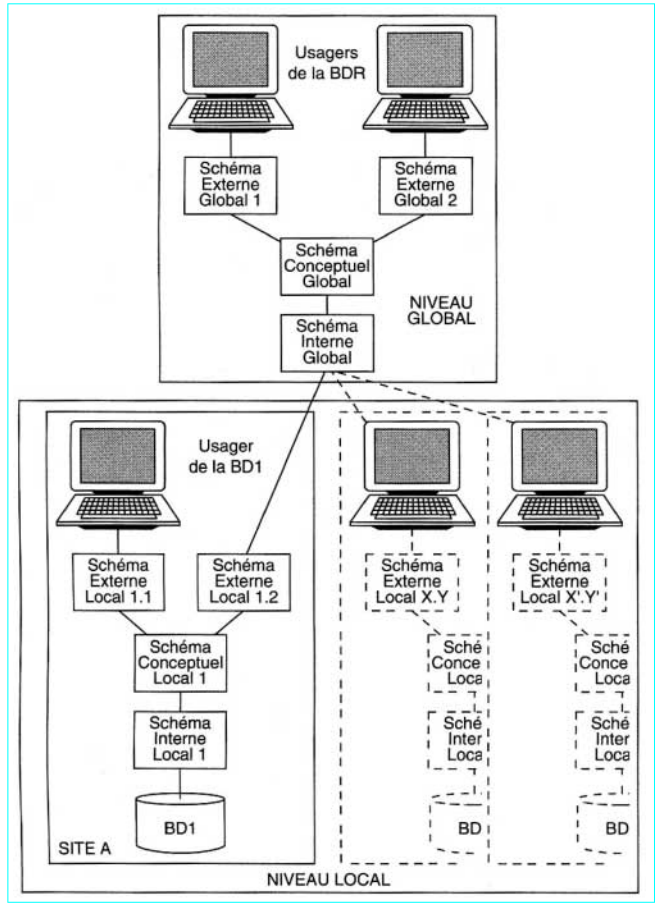


Figure 4 – Architecture d'une BDR selon la démarche descendante

La relation ASSURES peut être reconstituée à partir des fragments grâce aux opérations de jointure et d'union :

$$\text{ASSURES} = (\text{FR1} \bowtie \text{FR2}) \cup (\text{FR3} \bowtie \text{FR4})$$

où $\bowtie$ représente l'opération d'équijointure et $\cup$ l'opération d'union.

La reconstitution fait intervenir la clé primaire de la relation ASSURES, soit l'attribut NAS. Pour permettre une régénération sans perte d'information, des systèmes comme SDD-1 et R* associent à chaque n-uplet de la relation un identifiant de n-uplet. Il s'agit d'un identifiant interne géré par le système qui facilite les associations entre les relations dérivées (fragments) et la relation générique globale (ASSURES dans l'exemple).

3.1.2 Localisation des fragments

Dans la démarche descendante, la localisation des fragments est transparente pour l'utilisateur. Ainsi, la répartition des fragments dans les différents sites du réseau répond à des critères essentiellement techniques tels que le coût de stockage, le coût de mise à jour, la performance d'accès. Par exemple, chaque fois qu'un fragment est dupliqué dans le réseau, les applications peuvent accéder plus aisément à ce fragment mais, en contrepartie, le coût de mise à jour est plus élevé (modification des différentes copies).

Les SGBDR proposent généralement des modèles d'allocation des données qui sont paramétrables en fonction des contraintes liées à l'application (taux de mise à jour, temps de réponse,...).

3.2 Démarche ascendante

L'intérêt de cette démarche est de constituer une BDR à partir de BD existantes. Évidemment, c'est dans le cadre d'une telle démarche que se pose plus particulièrement le problème de la coopération de BD hétérogènes. Par exemple, on peut supposer qu'une BDR permette de faire coopérer certaines banques de données publiques accessibles au travers du réseau Teletel et sur lesquelles, actuellement, aucun traitement réparti n'est possible.

L'architecture de la BDR, si elle suit les niveaux proposés par l'ANSI/SPARC, n'en est pas moins distincte de celle de la démarche descendante. En effet, les bases de données locales sont considérées comme des entités connues des utilisateurs ; le SGBDR leur offre la possibilité d'opérer sur des données appartenant à plusieurs bases distinctes (on parle aussi de coopération de BD ou de multibase).

Ainsi, au niveau conceptuel, la BDR est définie comme un ensemble de BD entre lesquelles peuvent être définies des associations et diverses contraintes d'intégrité. À ce niveau, la localisation des BD reste inconnue.

Au niveau externe, les vues peuvent faire apparaître, ou non, la multiplicité des BD, selon le souhait de l'utilisateur.

Le niveau interne n'autorise pas une allocation des données aussi fine que dans l'architecture précédente. En effet, les bases de données locales étant existantes, seule est permise une duplication de la totalité d'une base de données (figure 5).

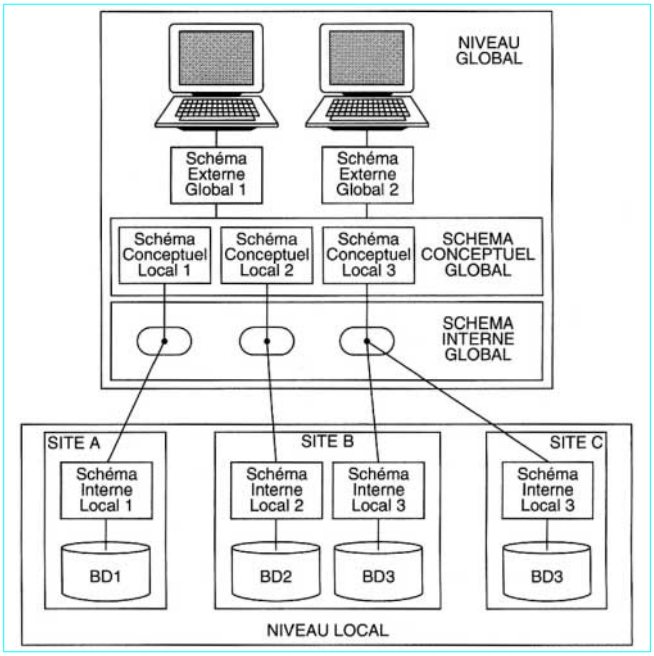


Figure 5 – Architecture d'une BDR selon la démarche ascendante

4. Traitement des requêtes

4.1 Principes

Dans un environnement réparti, les requêtes formulées à un niveau global sont décomposées en sous-requêtes. Ces sous-requêtes sont adressées aux systèmes disponibles sur les sites locaux où elles sont exécutées. Les réponses locales sont ensuite regroupées pour élaborer la réponse à la requête globale.

C'est ce processus que nous allons décrire en considérant des requêtes globales initialement formulées en SQL. Elles sont réécrites sous forme algébrique pour être réduites et optimisées. Le schéma de fragmentation permet de déterminer les requêtes locales adressées à chaque site. Les phases de ce processus sont visualisées sur la figure 6.

4.2 Stratégies de décomposition

Le traitement des requêtes dans un environnement réparti est plus complexe que dans un contexte classique car les paramètres affectant les performances sont plus nombreux. Il faut notamment considérer le coût des communications entre sites consécutivement à la fragmentation ou à la duplication des relations.

Pour présenter le processus de décomposition, nous ferons référence au schéma de base de données suivant :

ASSURES [NA, NOM, ADR, DPT]
CONTRATS [NCT, NA, DPT, TYPE, NIMM, BONUS]
SINISTRES [ND, NCT, DATE-SIN, EXPERT, MONTANT]

où les attributs ont la signification suivante :

NA	: code de l'assuré
NOM	: nom de l'assuré
ADR	: adresse de l'assuré
DPT	: département de l'assuré
NCT	: identifiant du contrat souscrit par l'assuré, l'assuré peut avoir souscrit plusieurs contrats
TYPE	: type de contrat ('TR' : tout risque, 'RAQVAM' : risques autres que véhicule à moteur,...)
NIMM	: numéro d'immatriculation
BONUS	: pourcentage de bonus/malus
ND	: numéro de dossier/sinistre
DATE-SIN	: date du sinistre
EXPERT	: nom de l'expert chargé de l'expertise
MONTANT	: montant estimé du sinistre par l'expert ('-'si ce montant n'est pas encore fixé).

La représentation algébrique fait intervenir la représentation symbolique suivante (nous ne donnons ici que les opérateurs de l'algèbre relationnelle utilisés par la suite) :

- projection : Π_X R projection de la relation R sur la liste d'attributs X ;
- sélection : σ_P R sélection des n-uplets de R vérifiant le prédicat P ;
- équijointure : R1 $\underset{A}{\bowtie}$ R2 jointure des relations R1 et R2 selon l'attribut A ;
  Le prédicat de jointure s'écrit R1.A = R2.A
- produit cartésien : $\times$ produit de deux relations ;
- union : $\cup$ union de deux relations ;
- intersection : $\cap$ intersection de deux relations.

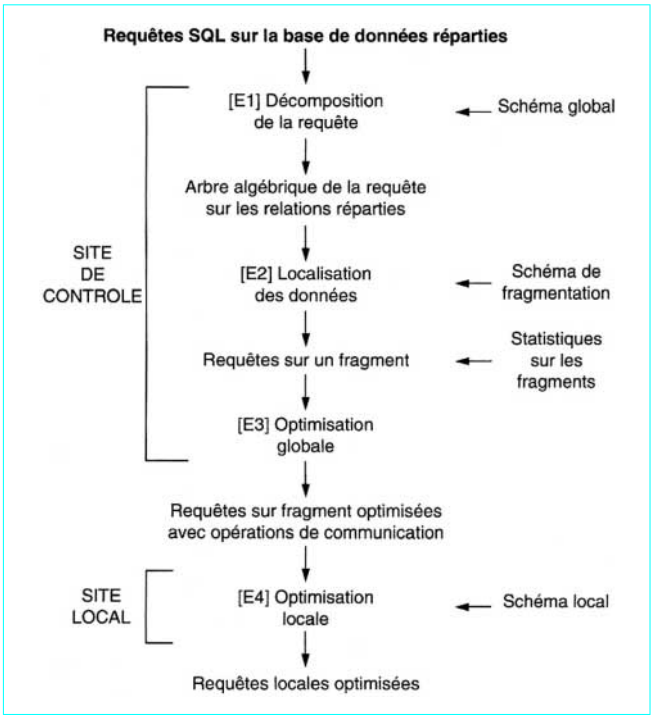
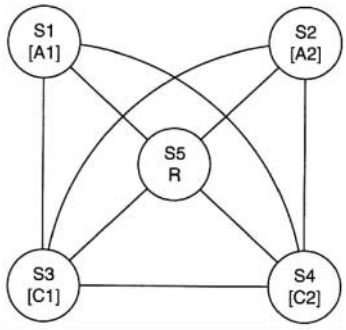


Figure 6 – Schéma général de traitement d'une requête répartie

Nous allons illustrer, avec un exemple simplifié, la problématique de la décomposition des requêtes. Supposons que nous soyons dans l'environnement réparti suivant :



- Cinq sites reliés entre eux : S1, S2, S3, S4, S5.
- Sur les quatre premiers sites sont répartis les quatre fragments : A1, A2, C1, C2 avec :

$A1 = \sigma_{DPT <= 31}$ ASSURES sur le site S1

$A2 = \sigma_{DPT > 31}$ ASSURES sur le site S2

$C1 = \sigma_{DPT <= 31}$ CONTRATS sur le site S3

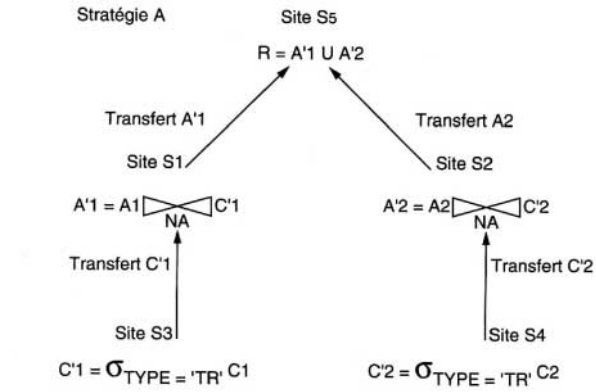
$C2 = \sigma_{DPT > 31}$ CONTRATS sur le site S4

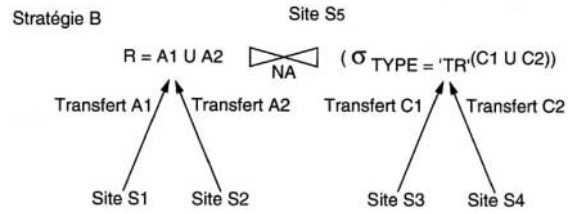
- La requête globale R est formulée sur le site S5. Le résultat de la requête est donc attendu sur ce site. Sa formulation à l'aide du langage SQL est la suivante :

```
select   NOM
from     ASSURES, CONTRATS
where    ASSURES.NA = CONTRATS.NA
and      TYPE = 'TR'
```

Cette requête permet de « Restituer le nom des assurés ayant souscrit un contrat "tout risque" ».

L'évaluation de cette requête peut conduire à de nombreuses décompositions algébriques mais nous ne retiendrons que les deux stratégies suivantes :





Nous allons déterminer intuitivement le coût de ces deux stratégies. Pour cela, on considère que :

— le coût d'accès a un n-uplet ta = 1 unité ;
— le coût de transfert d'un n-uplet tr = 10 unités ;
— la taille de A est de 300 n-uplets ;
— la taille de C est de 900 n-uplets (3 contrats en moyenne par assuré) ;
— 150 contrats sont des contrats « TR » ;
— les données sont uniformément distribuées sur les sites :
A1 = 150 n-uplets, A2 = 150 n-uplets,
C1 = 450 n-uplets, C2 = 450 n-uplets ;
— on dispose d'un index local sur NA pour A1, A2, C1, C2, ainsi que d'un index sur l'attribut TYPE pour C1 et C2.

- **Stratégie A**

— Coût de création de C'1 :

$$150^* \text{ ta} = 150$$

car il y a index sur un TYPE.

Coût de création de C'2 :

$$150^* \text{ ta} = 150$$

— Transfert de C'1 de S3 vers S1 et de C'2 de S4 vers S2 :

soit $(150^* \text{ tr}) \times 2 = 3\,000$

— Calcul de A. L'index sur NA est perdu lors du transfert (il existe sur C mais pas sur C'1). Pour réaliser la jointure, comme on a un index unique sur NA pour A, le coût de l'équijointure est de :

$$(150 \times 2 \times \text{ta}) \times 2 = 600$$

avec (1 accès à C + 1 accès à A)

— Transfert de A vers le site S5 :

$$600 \times \text{tr} = 6\,000$$

Coût total stratégie A = 300 + 3 000 + 600 + 6 000 = 9 900

● **Stratégie B**

— Transfert de A1 et A2 vers le site S5 :

$$300 \times tr = 3000$$

— Transfert de C1 et C2 vers le site S5 :

$$900 \times tr = 9000$$

— Calcul de la sélection TYPE = 'TR'. L'index sur TYPE étant perdu :

$$900 \times ta = 900$$

On génère un résultat à 150 éléments.

— Coût de calcul de la jointure (les index ayant été perdus lors du transfert) :

$$(300 \times 150) \times ta = 45\,000$$

Coût total stratégie B = 3 000 + 9 000 + 900 + 45 000 = 57 900

On a un rapport de l'ordre de 1 à 6 entre les coûts d'évaluation des stratégies A et B.

4.3 Étapes du traitement d'une requête répartie

L'utilisation d'une stratégie inadaptée peut accroître considérablement le coût d'évaluation d'une requête ; par conséquent, il est nécessaire de mettre en place un processus de décomposition/évaluation (reprenant les étapes mentionnées initialement) qui comprend essentiellement les phases suivantes :

[E1] Prétraitement de la requête avec :

1. normalisation de l'écriture de la requête,
2. analyse-vérification,
3. élimination de redondance,
4. réécriture ;

[E2] Localisation des données distribuées ;

[E3] Optimisation par calcul du coût et du temps de réponse ;

[E4] Optimisation locale (contexte centralisé).

L'étape [E1] a pour objectif l'élaboration d'un arbre algébrique optimisé d'évaluation de la requête.

La première phase [E1.1] consiste à transformer la requête pour faciliter son traitement ultérieur en la réécrivant sous forme conjonctive (ou disjonctive) normale en utilisant des règles de transformation.

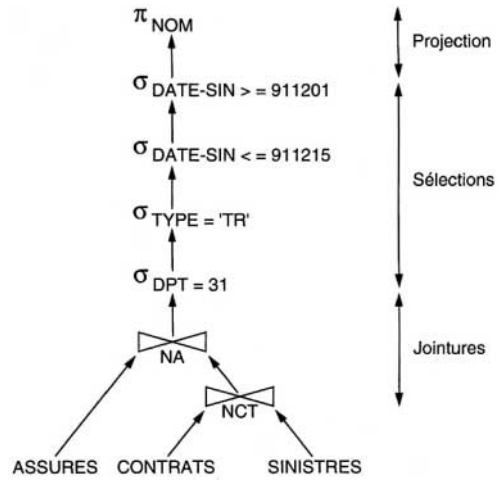
La phase [E1.2] a pour objet d'éliminer les requêtes normalisées pour lesquelles on détecte des anomalies ou des incohérences. Cette phase s'appuie sur la construction d'un graphe de requête et du graphe de jointure associé.

La phase [E1.3] élimine la redondance sémantique qui pourrait apparaître dans une requête faisant référence à des vues.

La dernière phase [E1.4] concerne la réécriture de la requête SQL en algèbre relationnelle. Soit la requête : « Restituer le nom des assurés de la Haute-Garonne (DPT = 31) ayant souscrit un contrat " tout risque " pour lequel on a enregistré un sinistre entre le 1er et le 15 décembre 1991 ».

```
select   NOM
from     ASSURES, CONTRATS, SINISTRES
where    ASSURES.NA = CONTRAT.NA
and      ASSURES.DPT = 31
and      CONTRAT.TYPE = "TR"
and      CONTRATS.NCT = SINISTRES.NCT
and      DATE-SIN >= 911201
and      DATE-SIN <= 911215
```

Un arbre algébrique correspondant à cette requête peut être le suivant :



Cet arbre fait ensuite l'objet d'une optimisation.

Les étapes [E2] et [E3] sont décrites dans les deux paragraphes qui suivent. L'étape [E4] concerne l'approche classique qui n'est pas reprise ici.

5. Localisation des données distribuées

Nous allons présenter maintenant la restructuration des requêtes en tenant compte, dans un contexte distribué, de la fragmentation de la base.

5.1 Fragmentation horizontale

Supposons que la relation ASSURES soit fragmentée horizontalement (fragmentation basée sur des opérations de sélection), comme suit :

A1 : $\sigma_{DPT < 31}$ (ASSURES)

A2 : $\sigma_{DPT = 31}$ (ASSURES)

A3 : $\sigma_{DPT > 31}$ (ASSURES)

ASSURES = A1 ∪ A2 ∪ A3

On réduit la requête en déterminant les relations intermédiaires de l'arbre algébrique de la requête qui seraient vides. Pour cela, on utilise des règles du type :

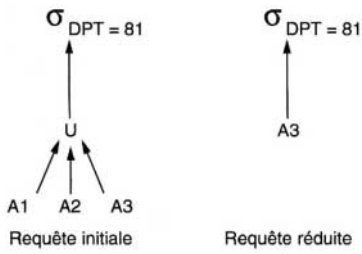
— soit R1,..., Rn les fragments avec Ri = σ_{Pi} (R)

alors σ_{Pk} (Ri) = ϕ si ∀ x ∈ Ri : (Pk (x) ∧ Pi (x))

Ainsi, si l'on écrit la requête :

```
select *
from ASSURES
where DPT = 81
```

On a la réduction suivante si l'on tient compte de la règle énoncée :



— si les prédicats de fragmentation pour Ri et Rj font intervenir les mêmes attributs, alors :

$$Ri \bowtie Rj = \phi \text{ si } \forall x \in Ri, \forall y \in Rj, \neg(Pi(x) \wedge Pj(x))$$

Supposons que CONTRATS soit fragmentée comme suit :

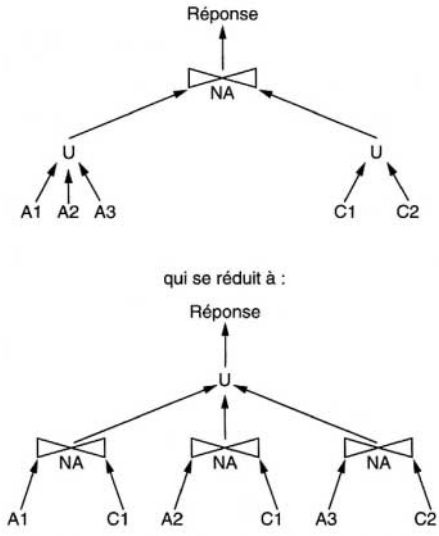
$C1 = \sigma_{DPT <= 31}$ CONTRATS

$C2 = \sigma_{DPT > 31}$ CONTRATS

La requête :

```
select *
from CONTRATS, ASSURES
where CONTRATS.NA = ASSURES.NA
```

conduit à l'arbre générique :



5.2 Fragmentation verticale

La reconstruction d'une relation s'effectue dans ce cas par jointure : supposons que ASSURES soit divisée en deux fragments verticaux :

$$A'1 = \pi_{NA,NOM}(ASSURES)$$

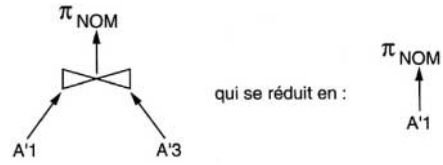
$$A'2 = \pi_{NA,ADR,DPT}(ASSURES)$$

$$A = A'1 \underset{NA}{\bowtie} A'2$$

Toute requête, comme dans le cas précédent, peut être ramenée à une forme réduite. Ainsi la requête :

```
select NOM from ASSURES
```

a pour forme générique :



Des techniques analogues sont mises en œuvre pour traiter les cas de fragmentation hybride. Soit :

$$A1 = \sigma_{NA < 1000}(\pi_{NA,NOM}(ASSURES))$$

$$A2 = \sigma_{NA > 1000}(\pi_{NA,NOM}(ASSURES))$$

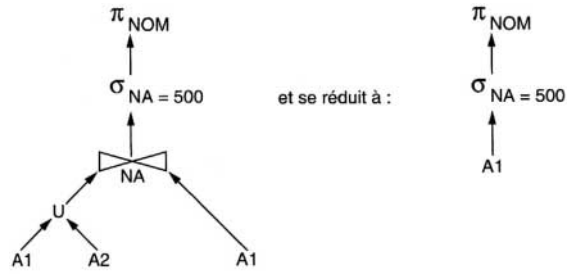
$$A3 = \pi_{NA,ADR,DPT}(ASSURES)$$

$$A = (A1 \cup A2) \underset{NA}{\bowtie} A3$$

La requête :

```
select NOM from ASSURES where NA = 500
```

a pour forme générique :



Le principe général des heuristiques de simplification consiste à :

— réaliser les sélections le plus tôt possible ;

— réaliser les projections le plus tard possible ;

— déterminer les prédicats en contradiction avec les prédicats de fragmentation horizontale pour éviter l'utilisation de ces fragments ;

— déterminer les fragments verticaux contenant les attributs utiles, les autres n'étant pas sollicités.

6. Optimisation de la stratégie d'exécution

Pour chaque arbre d'évaluation, le système détermine un coût selon une formule du type :

$$COUT = C_{UC} \cdot i + C_{ES} \cdot e + C_{MSG} \cdot m + C_{TR} \cdot t$$

avec

C_{UC}	coût d'exécution d'une instruction,
C_{ES}	coût d'une opération d'entrée/sortie,
C_{MSG}	coût d'initialisation d'un échange de messages,
C_{TR}	coût d'un transfert,
i	nombre d'instructions nécessaires à l'exécution de la requête,
e	nombre d'opérations d'entrées/sorties,
m	nombre d'échanges de messages,
t	taille des messages échangés.

Cette formule suggère que C_{TR} est constant, mais cette propriété n'est pas vérifiée avec des réseaux hétérogènes faisant intervenir des sous-réseaux dont les débits sont différents. Selon les caractéristiques des réseaux, on est amené à négliger soit le coût des opérations d'entrées/sorties (réseaux lents), soit le coût des communications (réseaux haut débit).

Le coût doit être différencié du temps de réponse donné par la formule :

$$\text{TEMPSREP} = C_{UC} \cdot M_i + C_{ES} \cdot M_e + C_{MSG} \cdot M_m + C_{TR} \cdot M_t$$

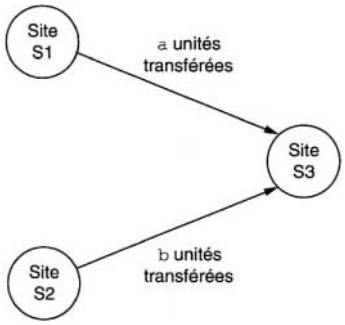
où M_e est le nombre maximal d'opérations d'entrées-sorties,

M_m est le nombre maximal d'échanges de messages,

M_t est la taille maximale des messages échangés,

M_i est le nombre maximal d'instructions à traiter séquentiellement pour l'exécution de la requête. Cet aspect fait référence à l'exécution d'instructions en parallèle.

Considérons l'**exemple** :



On suppose que C_{UC} et C_{ES} sont négligeables ainsi que i et e.

On suppose que C_{MSG} et C_{TR} sont exprimés en unités homogènes. Le coût des transferts vers le site S3 s'exprime ainsi :

$$\text{COUT} = 2 \cdot C_{MSG} + C_{TR} \cdot (a + b)$$

alors que le temps de réponse intégrant la possibilité de réalisation des deux transferts en parallèle sera :

$$\text{TEMPSREP} = \text{MAX}\,(C_{MSG} + C_{TR} \cdot a,\ C_{MSG} + C_{TR} \cdot b)$$

On minimise le temps de réponse avec un haut degré de parallélisme.

Pour réduire le temps de transfert, on utilise :

— le **concept de replication**, qui consiste à disposer de copies du même fragment sur plusieurs sites. On perd de l'espace mémoire, mais toute référence à un fragment copié ne nécessite pas son transfert. Ces copies sont gérées selon la technique ROWA (*Read Once Write Always*) qui indique qu'en lecture on fait référence à la copie locale, mais qu'une mise à jour doit être propagée sur tous les fragments répliqués ;

— les **semi-jointures** qui ont pour effet de minimiser la taille des relations intermédiaires nécessaires à l'évaluation de la requête, mais leur emploi ne doit pas être systématique et dépend du contexte.

La plupart des modèles développés essaient de déterminer l'arbre d'évaluation d'une requête qui minimise à la fois le coût global et le temps de réponse. Il existe de nombreux algorithmes d'optimisation détaillés dans [5].

7. Protocole de validation en deux phases

Dans un contexte distribué, la validation d'une requête globale qui se décompose en sous-requêtes locales, nécessite la mise en œuvre d'un protocole de validation en deux phases (*Two Phase Commit Protocol* ou protocole 2PC).

Un **module de contrôle global** (MCG) supervise l'exécution de la requête et détermine si cette dernière s'est terminée correctement. Pour cela, ce module de contrôle communique avec des **modules de contrôles locaux** (MCL), spécifiques à chaque site où sont exécutées les sous-requêtes. Selon que les sous-requêtes se sont bien exécutées ou non, le MCG peut ou non signaler la fin correcte de la requête à l'utilisateur.

Le pseudo-code qui suit précise le fonctionnement de ces modules. Une exécution correcte de la requête est validée par une opération de validation (COMMIT) qui rend effectives les modifications opérées sur la base. Une exécution incorrecte déclenche une opération de restauration (ROLLBACK) qui défait les modifications opérées par la requête et restaure la base de données dans un état cohérent antérieur.

Pseudo-code décrivant le fonctionnement du MCG

```
MCG
  REP-GLOBALE = "OK"
  FAIRE Pour chaque MCL TANT QUE REP-GLOBALE = "OK"
       REP-LOCALE = ""
       ENVOI-MSG "état d'achèvement de la sous-requête
                                          locale" au MCL ;
       Attente REP-LOCALE ou signal de dépassement de
                                                    temps
       SI REP-LOCALE <> "OK" ALORS
          REP-GLOBALE = '⅂OK'
       FSI
  FINFAIRE
   SI REP-GLOBALE = "OK" ALORS
    /* Délivrer la commande d'achèvement local COMMIT*/
    Forcer l'écriture de l'enregistrement point de
                                          validation
    "COMMIT" dans le journal du MCG (Commit Global) ;
    FAIRE pour chaque MCL
         REPETER JUSQU'À Accusé de réception positif
              Envoi demande COMMIT local au MCL ;
              Attente accusé de réception ou
                              dépassement de temps ;
         FIN REPETER
    FINFAIRE
   SINON
    /* Délivrer la commande d'achèvement local
                                          ROLLBACK */
    Forcer l'écriture de l'enregistrement point de
                                          validation
    "ROLLBACK" dans le journal du MCG (Rollback
                                          Global) ;
    FAIRE pour chaque MCL
         REPETER JUSQU'À Accusé de réception positif
                Envoi Demande ROLLBACK local au MCL ;
                Attente accusé de réception ou
                              dépassement de temps ;
         FIN REPETER
    FINFAIRE
   FSI
FIN-MCG
```

Pseudo-code décrivant le fonctionnement du MCL

```
MCL
   Attente du message "État d'achèvement de la
                                  sous-requête locale"
   Forcer l'écriture dans le journal local des
                                     informations pour
   Défaire/Refaire la sous-requête traitée
   Forcer l'écriture de l'enregistrement "Attente
       agrément global pour terminer" dans le journal
       local
   Si Exécution de la sous-requête correcte
      ALORS Envoi "OK" au MCG via la variable
                                              REP-LOCALE
      SINON Envoi "¬OK" au MCG ;
   FSI
   Attente de la demande globale délivrée par le MCG ;
   SI demande = "COMMIT"
      ALORS valider les modifications locales (COMMIT
                                               local)
   FSI ;
   SI demande = "ROLLBACK"
      ALORS Défaire les modifications locales
                                     (ROLLBACK local)
   FSI ;
   Libérer les ressources locales ;
   Envoi de l'accusé de réception au MCG ;
FIN MCL
```

Lors du fonctionnement du MCG, une défaillance peut survenir :

— avant création de l'enregistrement point de validation, auquel cas la procédure de redémarrage du MCG délivre un ROLLBACK à tous les MCL ;

— après création de l'enregistrement point de validation, auquel cas la procédure de redémarrage du MCG délivre soit un « COMMIT », soit un « ROLLBACK » aux MCL selon la nature de l'enregistrement point de validation.

Au niveau des MCL, les ressources mobilisées pour l'exécution de la requête globale doivent rester bloquées jusqu'à la terminaison de celle-ci. Lors du fonctionnement d'un MCL, si une anomalie survient :

— avant l'écriture de l'enregistrement « Attente d'agrément global pour terminer », la procédure de reprise locale renvoie « ¬OK » au MCG ce qui correspond à l'interprétation qu'a pu faire le MCG s'il y a eu dépassement de temps ;

— après l'écriture de l'enregistrement « Attente agrément local pour terminer », la procédure de reprise locale demande au MCG de renvoyer son message de demande d'achèvement local (soit par COMMIT, soit par ROLLBACK).

Les systèmes actuellement commercialisés prennent en compte à des degrés divers le concept de « transactions distribuées » nécessitant la mise en œuvre du protocole 2PC.

Certains imposent à l'utilisateur d'avoir une **seule base ouverte à un instant donné**, ce qui le conduit à contrôler lui-même l'exécution sérielle des sous-requêtes d'une requête globale. Nous n'avons pas ici ce que nous avons déjà désigné par le terme « Transaction distribuée ». C'est ce contexte que l'on trouve par exemple avec le système ORACLE/V6.

Certains systèmes, comme SYBASE, disposent des primitives de base du protocole 2PC mais demandent à l'utilisateur de coordonner les opérations de validation par COMMIT.

D'autres systèmes, comme INGRES ORACLE/V7, offrent le protocole complet.

8. Éléments de choix

Si l'on peut identifier les objectifs visés par les systèmes de gestion de bases de données réparties et les fonctionnalités permettant de les atteindre, il n'existe pas à l'heure actuelle de normes. Cela empêche une comparaison rationnelle de plusieurs systèmes entre eux à partir d'un référentiel stable qui serait joué par une telle norme.

En effet, si l'on s'intéresse à un ensemble de systèmes, ils n'intègrent généralement pas le même ensemble de fonctionnalités ou de services. Cela empêche une évaluation globale : on peut éventuellement déterminer un degré de complétude par rapport à une liste de fonctionnalités déterminées, ou faire une comparaison en ciblant une fonctionnalité particulière.

Par ailleurs, la mise en œuvre d'applications réparties utilisant des systèmes de gestion de bases de données ne nécessitent pas le recours *a priori* à un SGBD réparti. De nombreuses stratégies de développement de telles applications sont envisageables. Ces stratégies doivent tenir compte d'un environnement existant (homogène, hétérogène, multibases de données, intégration d'applications existantes déjà développées avec le SGBD « X »,...) qui généralement impose le SGBDR.

Un des modèles d'architecture s'imposant actuellement est l'**architecture client/serveur** qui ne peut être vue que comme un des supports possibles associé à la fonction de répartition.

Compte tenu de ces remarques, il apparaît difficile de proposer une liste objective de critères pour positionner les SGBD répartis les uns par rapport aux autres. Nous mentionnerons uniquement l'aspect fondamental qui est la présence ou l'absence du **protocole de validation en deux phases**.

Quelques systèmes relationnels et universels (non spécifiques d'un matériel et d'un système) intègrent le concept de répartition :

— INGRES et SYBASE proposent le protocole de validation deux phases ;

— ORACLE propose un environnement réparti dans la version 6 et doit intégrer le protocole 2PC dans la version 7.

La plupart des SGBD spécifiques à un constructeur de matériels intègrent, à des degrés divers, le concept de répartition. Nous pouvons citer R* qui est l'extension répartie de System-R (IBM) mais qui demeure un prototype.

9. Conclusion

Les bases de données réparties sont une solution séduisante pour parvenir à maîtriser la distribution des ressources informatiques sur plusieurs processeurs interconnectés.

Actuellement, les systèmes commerciaux offrent certaines fonctionnalités sans atteindre toujours la transparence à la répartition souhaitée par l'utilisateur.

Mais, au-delà de la complexité des techniques mises en œuvre et des performances qui se révèlent parfois décevantes (comparées aux temps de réponse de certains systèmes centralisés), d'autres écueils freinent le développement des bases de données réparties.

En effet, l'hétérogénéité actuelle et le manque de compatibilité des matériels et des logiciels au sein d'une même entreprise rendent particulièrement délicate la mise en place d'un SGBDR. Par conséquent, l'avènement d'un tel système ne peut être qu'un processus long et progressif qui devrait débuter par une prise de conscience collective des différents acteurs de l'entreprise de la nécessité d'un tel système (politique d'homogénéisation des matériels et logiciels, mise en place d'un réseau de communication fiable, répartition des responsabilités,...).

Un autre facteur est d'ordre psychologique. Responsabiliser un utilisateur par la gestion d'une base locale peut être un objectif rapidement atteint ; mais convaincre ce même utilisateur de partager ses données, même sous son contrôle, s'avère plus difficile à mettre en œuvre (et ce d'autant plus qu'il s'agit d'un utilisateur actuel de l'informatique individuelle).

Sur un plan purement technique, les systèmes de bases de données réparties devraient rapidement bénéficier des apports de la technologie objet. Cet aspect pourrait favoriser l'émergence de tels systèmes pour gérer les bases de données multimédia qui, de par leur volume et la complexité de leurs structures de données et de traitements, sont des candidates idéales à la répartition.

Références bibliographiques

[1] CHRISMENT (C.), PUJOLLE (G.) et ZURFLUH (G.). – *Langages de bases de données SQL et les évolutions vers l'objet.* H3128, traité Informatique. Techniques de l'Ingénieur, août 1999.

[2] *Heterogeneous Distributed Database Systems.* Computer – Special Issue, Vol. *24,* N° 12, ISSN 0018-9162, déc. 1991.

[3] DATE (C.J.). – *An introduction to database systems : Distributed Databases.* pp. 291-340 ; Addison Wesley Publishing Company, vol. *2,* ISBN 0-201-14474, 3 juil. 1985.

[4] GARDARIN (G.) et VALDURIEZ (P.). – *Relational Databases and Knowledge Bases : Distributed Databases,* pp. 413-434. Addison Wesley Publishing Company, ISBN 0-201-09955-1 (1989).

[5] OZSU (T.) et VALDURIEZ (P.). – *Distributed Database systems : Where are we now ?* IEEE Computer, pp. 68-78. ISSN 0018-9162, août 1991.

[6] ROTHNIE (J.B.) et al. – *Introduction to a system for distributed Databases (SDD-1).* ACM TODS, N° 5, mars 1980.

[7] WILLIAMS (R.) et al. – *R* : An overview of the Architecture.* IBM Research Report – RJ 3335, déc. 1981.

TECHNIQUES DE L'INGÉNIEUR

GAGNEZ DU TEMPS ET SÉCURISEZ VOS PROJETS EN UTILISANT UNE SOURCE ACTUALISÉE ET FIABLE

Techniques de l'Ingénieur propose la plus importante collection documentaire technique et scientifique en français !

Grâce à vos droits d'accès, retrouvez l'ensemble des **articles et fiches pratiques de votre offre, leurs compléments et mises à jour,** et bénéficiez des **services inclus.**

RÉDIGÉE ET VALIDÉE PAR DES EXPERTS	MISE À JOUR PERMANENTE	100 % COMPATIBLE SUR TOUS SUPPORTS NUMÉRIQUES	SERVICES INCLUS DANS CHAQUE OFFRE

- **+ de 350 000 utilisateurs**
- **+ de 10 000 articles de référence**
- **+ de 80 offres**
- **15 domaines d'expertise**

- **Automatique - Robotique**
- **Biomédical - Pharma**
- **Construction et travaux publics**
- **Électronique - Photonique**
- **Énergies**
- **Environnement - Sécurité**
- **Génie industriel**
- **Ingénierie des transports**
- **Innovation**
- **Matériaux**
- **Mécanique**
- **Mesures - Analyses**
- **Procédés chimie - Bio - Agro**
- **Sciences fondamentales**
- **Technologies de l'information**

Pour des offres toujours plus adaptées à votre métier, découvrez les offres dédiées à votre secteur d'activité

Depuis plus de 70 ans, Techniques de l'Ingénieur est la source d'informations de référence des bureaux d'études, de la R&D et de l'innovation.

www.techniques-ingenieur.fr

CONTACT : Tél. : + 33 (0)1 53 35 20 20 - Fax : +33 (0)1 53 26 79 18 - E-mail : infos.clients@teching.com

TECHNIQUES DE L'INGÉNIEUR

LES AVANTAGES ET SERVICES compris dans les offres Techniques de l'Ingénieur

ACCÈS

Accès illimité aux articles en HTML
Enrichis et mis à jour pendant toute la durée de la souscription

Téléchargement des articles au format PDF
Pour un usage en toute liberté

Consultation sur tous les supports numériques
Des contenus optimisés pour ordinateurs, tablettes et mobiles

SERVICES ET OUTILS PRATIQUES

Questions aux experts*
Les meilleurs experts techniques et scientifiques vous répondent

Articles Découverte
La possibilité de consulter des articles en dehors de votre offre

Dictionnaire technique multilingue
45 000 termes en français, anglais, espagnol et allemand

Archives
Technologies anciennes et versions antérieures des articles

Impression à la demande
Commandez les éditions papier de vos ressources documentaires

Alertes actualisations
Recevez par email toutes les nouveautés de vos ressources documentaires

*Questions aux experts est un service réservé aux entreprises, non proposé dans les offres écoles, universités ou pour tout autre organisme de formation.

ILS NOUS FONT CONFIANCE

AgroParisTech · AIRBUS · cea · CESI Enseignement Supérieur et Formation Professionnelle · cnes Centre National d'Études Spatiales · CSTB le futur en construction · ENGIE · GE · L'ORÉAL · Leroy-Somer · École Polytechnique Université Paris-Saclay · SAFRAN Aerospace Defence Security · Sagemcom · SAINT-GOBAIN · Schneider Electric · THALES · TOTAL · Valeo · VEOLIA · Wallonie · SPW Service public de Wallonie

www.techniques-ingenieur.fr

CONTACT : Tél. : + 33 (0)1 53 35 20 20 - Fax : +33 (0)1 53 26 79 18 - E-mail : infos.clients@teching.com