PARTIE

6 SGBD distribués et réplication

CHAPITRE

24 SGBD distribués—Concepts et Conception

objectifs du chapitre

Dans ce chapitre, vous apprendrez :

- The need for distributed databases.
- The differences between distributed DBMSs, distributed processing, and parallel DBMSs.
- The advantages and disadvantages of distributed DBMSs.
- The problems of heterogeneity in a distributed DBMS.
- Basic networking concepts.
- The functions that should be provided by a distributed DBMS.
- An architecture for a distributed DBMS.
- The main issues associated with distributed database design: fragmentation, replication, and allocation.
- How fragmentation should be carried out.
- The importance of allocation and replication in distributed databases.
- The levels of transparency that should be provided by a distributed DBMS.
- Comparison criteria for distributed DBMSs.

La technologie des bases de données nous a fait passer d'un paradigme de traitement des données dans lequel chaque application définissait et maintenait ses propres données à un paradigme dans lequel les données étaient définies et administrées de manière centralisée. Ces derniers temps, nous avons assisté aux développements rapides des technologies de réseau et de communication de données, incarnés par Internet, l'informatique mobile et sans fil, les appareils intelligents et l'informatique en grille. Maintenant, avec la combinaison de ces deux technologies, la technologie des bases de données distribuées peut changer le mode de travail de centralisé à décentralisé. Cette technologie combinée est l'un des développements majeurs dans le domaine des systèmes de bases de données.

Dans les chapitres précédents, nous nous sommes concentrés sur les systèmes de bases de données centralisées, c'est-à-dire les systèmes avec une seule base de données logique située sur un site sous le contrôle d'un seul SGBD. Dans ce chapitre, nous abordons les concepts et les problèmes du système de gestion de base de données distribuée (DDBMS), qui permet aux utilisateurs d'accéder non seulement aux

des données sur leur propre site mais également des données stockées sur des sites distants. Certains prétendent que les SGBD centralisés finiront par devenir une «ÿcuriosité antiqueÿ» à mesure que les organisations évolueront vers des SGBD distribués.

Structure de ce chapitre Dans la section 24.1, nous introduisons les concepts de base du DDBMS et faisons des distinctions entre les DDBMS, le traitement distribué et les SGBD parallèles. Dans la section 24.2, nous fournissons une très brève introduction au réseautage pour aider à clarifier certaines des questions dont nous discuterons plus tard. Dans la section 24.3, nous examinons les fonctionnalités étendues que nous attendons d'être fournies par un DDBMS. Nous examinons également les architectures de référence possibles pour un DDBMS en tant qu'extensions de l'architecture ANSISPARC présentée au chapitre 2. Dans la section 24.4, nous discutons de la façon d'étendre la méthodologie de conception de base de données présentée dans la partie 4 de ce livre pour tenir compte de la distribution des données. Dans la section 24.5, nous discutons des transparences que nous nous attendrions à trouver dans un DDBMS et concluons dans la section 24.6 avec un bref examen des douze règles de Date pour un DDBMS. Les exemples de ce chapitre sont à nouveau tirés de l'étude de cas Dream Home décrite à la section 11.4 et à l'annexe A.

Pour l'avenir, dans le chapitre suivant, nous examinons comment les protocoles de contrôle de la concurrence, de gestion des interblocages et de contrôle de la récupération dont nous avons parlé au chapitre 22 peuvent être étendus pour répondre à l'environnement distribué. Au chapitre 26, nous discutons du serveur de réplication, qui est une approche alternative et potentiellement plus simplifiée de la distribution des données et des bases de données mobiles. Nous examinons également comment Oracle prend en charge la réplication et la mobilité des données.

24.1 Présentation

Une motivation majeure derrière le développement de systèmes de bases de données est le désir d'intégrer les données opérationnelles d'une organisation et de fournir un accès contrôlé aux données. Bien que l'intégration et l'accès contrôlé puissent impliquer une centralisation, ce n'est pas l'intention. En effet, le développement des réseaux informatiques favorise un mode de travail décentralisé. Cette approche décentralisée reflète la structure organisationnelle de nombreuses entreprises, qui sont réparties logiquement en divisions, départements, projets, etc., et physiquement réparties en bureaux, usines, usines, où chaque unité conserve ses propres données opérationnelles (Date, 2000) . La possibilité de partager les données et l'efficacité de l'accès aux données devraient être améliorées par le développement d'un système de base de données distribué qui reflète cette structure organisationnelle, rend les données de toutes les unités accessibles et stocke les données à proximité de l'endroit où elles sont le plus fréquemment utilisées.

Les SGBD distribués devraient aider à résoudre le problème des îlots d'information. Les bases de données sont parfois considérées comme des îlots électroniques qui sont des lieux distincts et généralement inaccessibles, comme des îles éloignées. Cela peut être le résultat d'une séparation géographique, d'architectures informatiques incompatibles, de protocoles de communication incompatibles, etc. L'intégration des bases de données dans un tout logique peut empêcher cette façon de penser.

24.1.1 Concepts Pour

commencer la discussion sur les SGBD distribués, donnons d'abord quelques définitions.

Distribué base de données Une collection logiquement interdépendante de données partagées (et une description de ces données) physiquement distribuée sur un réseau informatique.

Distribué SGBD Le système logiciel qui permet la gestion de la base de données distribuée et rend la distribution transparente pour les utilisateurs.

Un système de gestion de bases de données distribuées (DDBMS) consiste en une seule base de données logique divisée en plusieurs fragments. Chaque fragment est stocké sur un ou plusieurs ordinateurs sous le contrôle d'un SGBD séparé, les ordinateurs étant connectés par un réseau de communication. Chaque site est capable de traiter indépendamment les demandes des utilisateurs qui nécessitent un accès aux données locales (c'est-à-dire que chaque site dispose d'un certain degré d'autonomie locale) et est également capable de traiter les données stockées sur d'autres ordinateurs du réseau.

Les utilisateurs accèdent à la base de données distribuée via des applications, qui sont classées comme celles qui ne nécessitent pas de données d'autres sites (applications locales) et celles qui nécessitent des données d'autres sites (applications globales). Nous exigeons qu'un DDBMS ait au moins une application globale. Un DDBMS a donc les caractéristiques suivantes :

- une collection de données partagées logiquement
- liéesÿ; les données sont divisées en plusieurs
- fragmentsÿ; les fragments peuvent être répliquésÿ;
- les fragments/répliques sont attribués aux sitesÿ; les
- sites sont reliés par un réseau de communication ; les
- données de chaque site sont sous le contrôle d'un SGBDÿ; le
- SGBD de chaque site peut gérer les applications locales, de manière autonome ;
- chaque SGBD participe à au moins une application globale.

Il n'est pas nécessaire que chaque site du système ait sa propre base de données locale, comme l'illustre la topologie du DDBMS illustrée à la figure 24.1.

EXEMPLE 24.1 DreamHome

À l'aide de la technologie des bases de données distribuées, DreamHome peut mettre en œuvre son système de base de données sur un certain nombre de systèmes informatiques distincts plutôt que sur un seul ordinateur principal centralisé. Les systèmes informatiques peuvent être situés dans chaque succursale locale : par exemple, Londres, Aberdeen et Glasgow. Un réseau reliant les ordinateurs permettra aux succursales de communiquer entre elles et un DDBMS leur permettra d'accéder aux données stockées dans une autre succursale. Ainsi, un client vivant à Glasgow peut se rendre à la succursale la plus proche pour connaître les propriétés disponibles à Londres, plutôt que d'avoir à téléphoner ou écrire à la succursale de Londres pour plus de détails.

Alternativement, si chaque succursale DreamHome possède déjà sa propre base de données (disparate), un DDBMS peut être utilisé pour intégrer les bases de données séparées dans une seule base de données logique, rendant à nouveau les données locales plus largement disponibles.

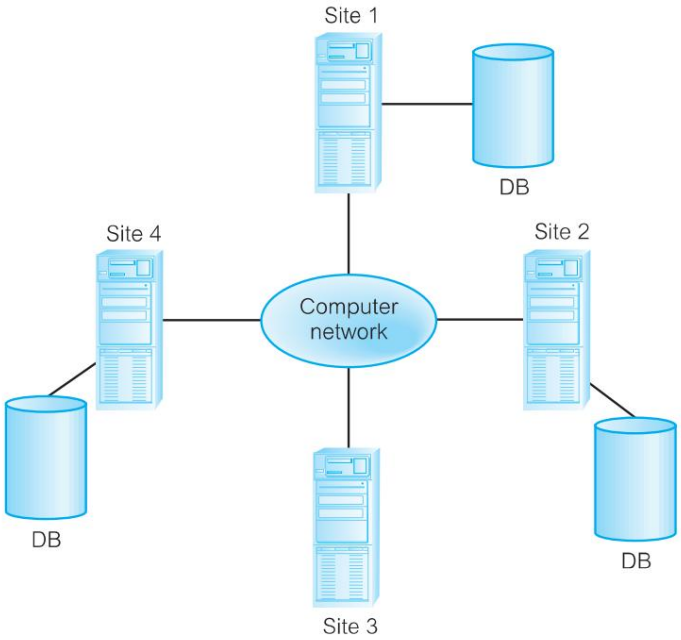

Illustration 24.1 Distributed database management system.

D'après la définition du DDBMS, le système est censé rendre la distribution transparente (invisible) pour l'utilisateur. Ainsi, le fait qu'une base de données distribuée soit découpée en fragments pouvant être stockés sur différents ordinateurs et peut-être répliqué, doit être caché à l'utilisateur. L'objectif de la transparence est de faire apparaître le système distribué comme un système centralisé. C'est parfois appelé le principe fondamental des SGBD distribués (Date, 1987b). Cette exigence fournit des fonctionnalités importantes pour l'utilisateur final, mais crée malheureusement de nombreux problèmes supplémentaires qui doivent être gérés par le DDBMS, comme nous le verrons dans la section 24.5.

Traitement distribué

Il est important de faire la distinction entre un SGBD distribué et un En traitement.

Distribué En traitement	Une base de données centralisée accessible via un réseau informatique.

Le point clé de la définition d'un SGBD distribué est que le système se compose de données qui sont physiquement distribuées sur un certain nombre de sites du réseau. Si les données sont centralisées, même si d'autres utilisateurs peuvent accéder aux données sur le réseau, nous ne considérons pas qu'il s'agit d'un SGBD distribué mais simplement d'un En traitement. Nous illustrons la topologie du traitement distribué dans la figure 24.2.

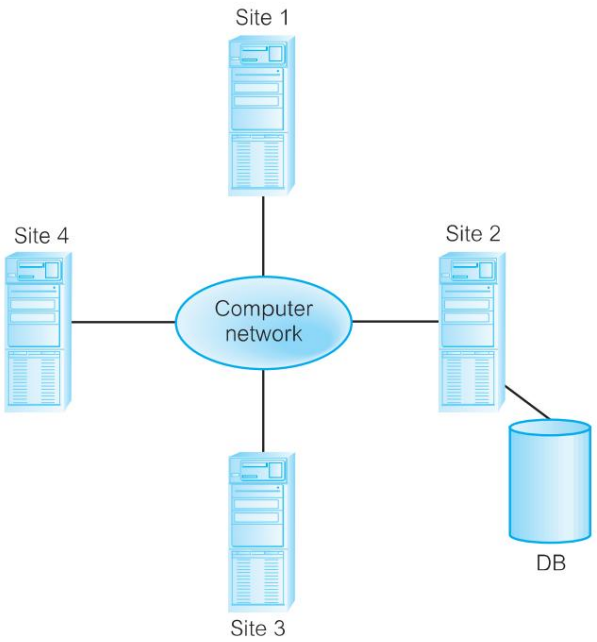

Figure 24.2
Distributed processing.

Comparez cette figure, qui a une base de données centrale sur le site 2, avec la Figure 24.1, qui montre plusieurs sites chacun avec sa propre base de données (DB).

SGBD parallèles

Nous faisons également une distinction entre un SGBD distribué et un SGBD parallèle.

SGBD parallèle — SGBD s'exécutant sur plusieurs processeurs et disques et conçu pour exécuter des opérations en parallèle, dans la mesure du possible, afin d'améliorer les performances.

Les SGBD parallèles sont à nouveau basés sur le principe que les systèmes à processeur unique ne peuvent plus répondre aux exigences croissantes d'évolutivité, de fiabilité et de performances rentables. Une alternative puissante et financièrement attrayante à un SGBD piloté par un seul processeur est un SGBD parallèle piloté par plusieurs processeurs. Les SGBD parallèles relient plusieurs machines plus petites pour obtenir le même débit qu'une seule machine plus grande, souvent avec une évolutivité et une fiabilité supérieures à celles des SGBD à processeur unique.

Pour fournir à plusieurs processeurs un accès commun à une seule base de données, un SGBD parallèle doit permettre une gestion partagée des ressources. Les ressources partagées et la façon dont ces ressources partagées sont mises en œuvre affectent directement les performances et l'évolutivité du système, qui à leur tour déterminent sa pertinence pour une application/un environnement donné. Les trois architectures principales pour les SGBD parallèles, illustrées à la Figure 24.3, sontÿ:

- la memoire partagée;
- disque partagéÿ; ne
- partageait rien.

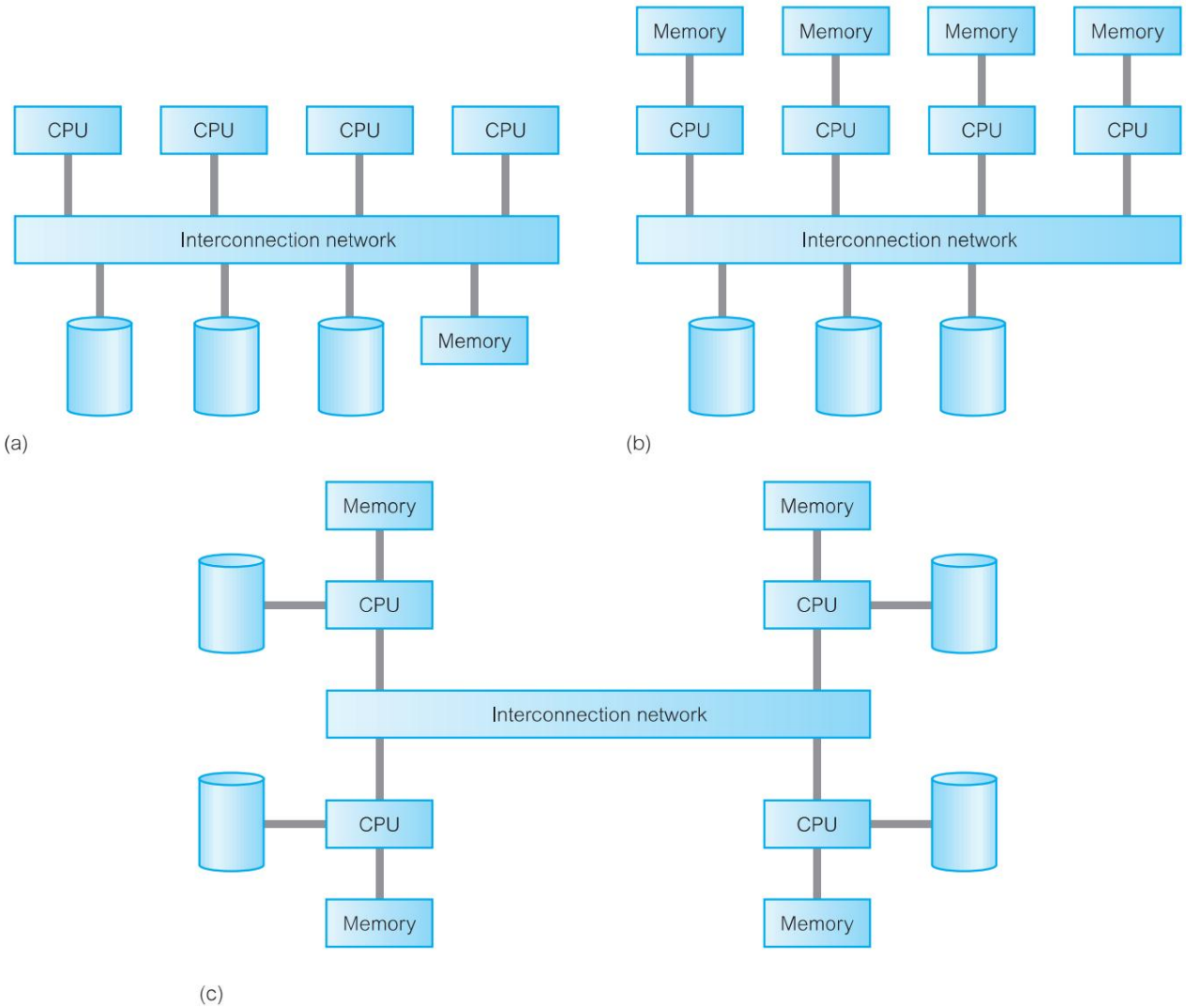

Figure 24.3 Parallel database architectures: (a) shared memory; (b) shared disk; (c) shared nothing.

La mémoire partagée est une architecture étroitement couplée dans laquelle plusieurs processeurs d'un même système partagent la mémoire système. Connue sous le nom de multitraitement symétrique (SMP), cette approche est devenue populaire sur des plates-formes allant des postes de travail personnels qui prennent en charge quelques microprocesseurs en parallèle, aux grandes machines basées sur RISC (Reduced Instruction Set Computer), jusqu'aux plus grands châssis principaux. Cette architecture fournit un accès aux données à haut débit pour un nombre limité de processeurs, mais elle n'est pas évolutive au-delà d'environ 64 processeurs, auquel cas le
réseau d'interconnexion devient un goulot d'étranglement.

Le disque partagé est une architecture faiblement couplée optimisée pour les applications qui sont par nature centralisées et qui nécessitent une disponibilité et des performances élevées. Chaque processeur peut accéder directement à tous les disques, mais chacun possède sa propre mémoire privée. Comme le partagé

aucune architecture, l'architecture de disque partagé élimine le goulot d'étranglement des performances de la mémoire partagée. Contrairement à l'architecture sans partage, cependant, l'architecture de disque partagé élimine ce goulot d'étranglement sans introduire la surcharge associée aux données partitionnées physiquement. Les systèmes de disques partagés sont parfois appelés clusters.

Le partage sans partage, souvent appelé traitement massivement parallèle (MPP), est une architecture multiprocesseur dans laquelle chaque processeur fait partie d'un système complet, avec sa propre mémoire et son propre stockage sur disque. La base de données est partitionnée entre tous les disques de chaque système associé à la base de données, et les données sont disponibles de manière transparente pour les utilisateurs sur tous les systèmes. Cette architecture est plus évolutive que la mémoire partagée et peut facilement supporter un grand nombre de processeurs. Cependant, les performances ne sont optimales que lorsque les données demandées sont stockées localement.

Bien que la définition de l'absence de partage inclut parfois des SGBD distribués, la distribution des données dans un SGBD parallèle repose uniquement sur des considérations de performances. De plus, les nœuds d'un SGBD sont généralement répartis géographiquement, administrés séparément et ont un réseau d'interconnexion plus lent, alors que les nœuds d'un SGBD parallèle se trouvent généralement sur le même ordinateur ou sur le même site.

La technologie parallèle est généralement utilisée pour de très grandes bases de données, éventuellement de l'ordre de téraoctets (1012 octets), ou des systèmes qui doivent traiter des milliers de transactions par seconde. Ces systèmes ont besoin d'accéder à de gros volumes de données et doivent fournir des réponses rapides aux requêtes. Un SGBD parallèle peut utiliser l'architecture sous-jacente pour améliorer les performances de l'exécution de requêtes complexes à l'aide de techniques d'analyse, de jointure et de tri parallèles qui permettent à plusieurs nœuds de processeur de partager automatiquement la charge de travail de traitement. Nous abordons cette architecture plus en détail dans le chapitre 31 sur l'entreposage de données. Il suffit de noter ici que tous les principaux fournisseurs de SGBD produisent des versions parallèles de leurs moteurs de base de données.

24.1.2 Avantages et inconvénients des DDBMS

La distribution des données et des applications présente des avantages potentiels par rapport aux systèmes de base de données centralisés traditionnels. Malheureusement, il y a aussi des inconvénients. Dans cette section, nous passons en revue les avantages et les inconvénients du DDBMS.

Avantages

Reflète la structure organisationnelle De nombreuses organisations sont naturellement réparties sur plusieurs sites. Par exemple, DreamHome a de nombreux bureaux dans différentes villes. Il est naturel que les bases de données utilisées dans une telle application soient réparties sur ces emplacements. DreamHome peut conserver une base de données dans chaque succursale contenant des détails tels que le personnel qui travaille à cet endroit, les propriétés à louer et les clients qui possèdent ou souhaitent louer ces propriétés. Le personnel d'une succursale effectuera des recherches locales dans la base de données. Le siège de la société peut souhaiter effectuer des enquêtes globales impliquant l'accès aux données dans toutes ou plusieurs succursales.

Amélioration de la partageabilité et de l'autonomie locale La distribution géographique d'une organisation peut se refléter dans la distribution des données ; les utilisateurs d'un site peuvent

accéder aux données stockées sur d'autres sites. Les données peuvent être placées sur le site à proximité des utilisateurs qui utilisent normalement ces données. De cette façon, les utilisateurs ont un contrôle local des données et ils peuvent par conséquent établir et appliquer des politiques locales concernant l'utilisation de ces données. Un DBA global est responsable de l'ensemble du système. Généralement, une partie de cette responsabilité est dévolue au niveau local, de sorte que le DBA local peut gérer le SGBD local (voir section 10.15).

Disponibilité améliorée Dans un SGBD centralisé, une panne informatique met fin aux opérations du SGBD. Cependant, une panne sur un site d'un SGBDR ou une panne d'un lien de communication rendant certains sites inaccessibles ne rend pas tout le système inopérant. Les SGBD distribués sont conçus pour continuer à fonctionner malgré de tels échecs. Si un seul nœud échoue, le système peut être en mesure de rediriger les demandes du nœud défaillant vers un autre site.

Fiabilité améliorée Étant donné que les données peuvent être répliquées de sorte qu'elles existent sur plusieurs sites, la défaillance d'un nœud ou d'une liaison de communication ne rend pas nécessairement les données inaccessibles.

Performances améliorées Comme les données sont situées à proximité du site « le plus demandé » et compte tenu du parallélisme inhérent des SGBD distribués, la vitesse d'accès à la base de données peut être meilleure que celle réalisable à partir d'une base de données centralisée distante. De plus, étant donné que chaque site ne gère qu'une partie de l'ensemble de la base de données, il se peut qu'il n'y ait pas le même conflit pour les services CPU et I/O caractérisés par un SGBD centralisé.

Économie Dans les années 1960, la puissance de calcul était calculée selon le carré des coûts de l'équipement : trois fois le coût fournirait neuf fois la puissance. C'était ce qu'on appelait la loi de Grosch. Cependant, il est maintenant généralement admis qu'il en coûte beaucoup moins cher de créer un système d'ordinateurs plus petits avec la puissance équivalente d'un seul gros ordinateur. Cela rend plus rentable pour les divisions et les départements de l'entreprise d'obtenir des ordinateurs séparés. Il est également beaucoup plus rentable d'ajouter des postes de travail à un réseau que de mettre à jour un système central.

La deuxième économie de coût potentielle se produit lorsque les bases de données sont géographiquement éloignées et que les applications nécessitent un accès aux données distribuées. Dans de tels cas, en raison du coût relatif des données transmises sur le réseau par rapport au coût de l'accès local, il peut être beaucoup plus économique de partitionner l'application et d'effectuer le traitement localement sur chaque site.

Croissance modulaire Dans un environnement distribué, il est beaucoup plus facile de gérer l'expansion. De nouveaux sites peuvent être ajoutés au réseau sans affecter les opérations des autres sites. Cette flexibilité permet à une organisation de se développer relativement facilement. L'augmentation de la taille de la base de données peut généralement être gérée en ajoutant de la puissance de traitement et de stockage au réseau. Dans un SGBD centralisé, la croissance peut entraîner des modifications à la fois matérielles (l'acquisition d'un système plus puissant) et logicielles (l'acquisition d'un SGBD plus puissant ou plus configurable). La transition vers le nouveau matériel/logiciel pourrait donner lieu à de nombreuses difficultés.

Intégration Au début de cette section, nous avons noté que l'intégration était un avantage clé de l'approche SGBD, et non la centralisation. L'intégration des systèmes hérités est un exemple particulier qui montre comment certaines organisations sont obligées de s'appuyer sur le traitement de données distribué pour permettre à leurs systèmes hérités de coexister avec leurs systèmes plus modernes. Dans le même temps, aucun package ne peut fournir toutes les fonctionnalités dont une organisation a besoin de nos jours. Ainsi, il est important pour les organisations de pouvoir intégrer des composants logiciels de différents fournisseurs pour répondre à leurs besoins spécifiques.

Rester compétitif Il existe un certain nombre de développements relativement récents qui s'appuient fortement sur la technologie des bases de données distribuées telles que le commerce électronique, le travail collaboratif assisté par ordinateur et la gestion des flux de travail. De nombreuses entreprises ont dû réorganiser leurs activités et utiliser la technologie des bases de données distribuées pour rester compétitives. Par exemple, bien que plus de personnes ne loueront pas nécessairement des propriétés simplement parce qu'Internet existe, DreamHome peut perdre une partie de sa part de marché s'il ne permet pas aux clients de voir les propriétés en ligne maintenant.

Inconvénients

Complexité Un SGBD distribué qui cache la nature distribuée à l'utilisateur et fournit un niveau acceptable de performances, de fiabilité et de disponibilité est intrinsèquement plus complexe qu'un SGBD centralisé. Le fait que les données puissent être répliquées ajoute également un niveau supplémentaire de complexité au SGBD distribué. Si le logiciel ne gère pas correctement la réplication des données, il y aura une dégradation de la disponibilité, de la fiabilité et des performances par rapport au système centralisé, et les avantages que nous avons cités précédemment deviendront des inconvénients.

Coût Une complexité accrue signifie que nous pouvons nous attendre à ce que les coûts d'approvisionnement et de maintenance d'un SGBDR soient plus élevés que ceux d'un SGBD centralisé. De plus, un SGBD distribué nécessite du matériel supplémentaire pour établir un réseau entre les sites. Il y a des frais de communication permanents encourus avec l'utilisation de ce réseau. Il existe également des coûts de main-d'œuvre supplémentaires pour gérer et maintenir les SGBD locaux et le réseau sous-jacent.

Sécurité Dans un système centralisé, l'accès aux données peut être facilement contrôlé. Cependant, dans un SGBD distribué, non seulement l'accès aux données répliquées doit être contrôlé à plusieurs endroits, mais le réseau lui-même doit être sécurisé. Dans le passé, les réseaux étaient considérés comme un moyen de communication non sécurisé. Bien que cela soit encore partiellement vrai, des développements importants ont été réalisés pour rendre les réseaux plus sûrs.

Contrôle d'intégrité plus difficile L'intégrité de la base de données fait référence à la validité et à la cohérence des données stockées. L'intégrité est généralement exprimée en termes de contraintes, qui sont des règles de cohérence que la base de données n'est pas autorisée à violer. L'application de contraintes d'intégrité nécessite généralement l'accès à une grande quantité de données qui définissent la contrainte mais qui ne sont pas impliquées dans l'opération de mise à jour proprement dite. Dans un SGBD distribué, les coûts de communication et de traitement nécessaires pour appliquer les contraintes d'intégrité peuvent être prohibitifs. Nous revenons sur ce problème dans la section 25.4.5.

Absence de standards Bien que les SGBD distribués dépendent d'une communication efficace, nous commençons seulement à voir l'apparition d'une communication standard

TABLEAU 24.1 Summary of advantages and disadvantages of DDBMSs.

AVANTAGES	DÉSAVANTAGES
Reflects organizational structure	Complexity
Improved shareability and local autonomy	Cost
Improved availability	Security
Improved reliability	Integrity control more difficult
Improved performance	Lack of standards
Economics	Lack of experience
Modular growth	Database design more complex
Integration	
Remaining competitive	

et les protocoles d'accès aux données. Ce manque de normes a considérablement limité le potentiel des SGBD distribués. Il n'existe pas non plus d'outils ou de méthodologies pour aider les utilisateurs à convertir un SGBD centralisé en un SGBD distribué.

Manque d'expérience Les SGBD distribués à usage général n'ont pas été largement acceptés, bien que de nombreux protocoles et problèmes soient bien compris. Par conséquent, nous n'avons pas encore le même niveau d'expérience dans l'industrie que nous avons avec les SGBD centralisés. Pour un éventuel adopteur de cette technologie, cela peut être un élément dissuasif important.

Conception de base de données plus complexe Outre les difficultés normales de conception d'une base de données centralisée, la conception d'une base de données distribuée doit tenir compte de la fragmentation des données, de l'allocation des fragments à des sites spécifiques et de la réplication des données. Nous discutons de ces problèmes dans la section 24.4.

Les avantages et les inconvénients des DDBMS sont résumés dans le tableau 24.1.

24.1.3 DDBMS homogènes et hétérogènes

Un DDBMS peut être classé comme homogène ou hétérogène. Dans un système homogène, tous les sites utilisent le même produit SGBD. Dans un système hétérogène, les sites peuvent exécuter différents produits SGBD, qui n'ont pas besoin d'être basés sur le même modèle de données sous-jacent, et ainsi le système peut être composé de SGBD relationnels, réseau, hiérarchiques et orientés objet.

Les systèmes homogènes sont beaucoup plus faciles à concevoir et à gérer. Cette approche permet une croissance incrémentielle, facilitant l'ajout d'un nouveau site au DDBMS, et permet des performances accrues en exploitant la capacité de traitement parallèle de plusieurs sites.

Les systèmes hétérogènes se produisent généralement lorsque des sites individuels ont mis en œuvre leurs propres bases de données et que l'intégration est envisagée à un stade ultérieur. Dans un système hétérogène, des traductions sont nécessaires pour permettre la communication entre différents SGBD. Pour assurer la transparence du SGBD, les utilisateurs doivent pouvoir effectuer des requêtes dans la langue du SGBD sur leur site local. Le système a alors pour tâche de localiser les données et d'effectuer toute traduction nécessaire. Des données peuvent être requises d'un autre site qui peut avoirÿ:

- matériel différent;
- différents produits SGBDÿ;
- différents matériels et différents produits SGBD.

Si le matériel est différent mais que les produits SGBD sont les mêmes, la traduction est simple, impliquant le changement de codes et de longueurs de mots. Si les produits SGBD sont différents, la traduction est compliquée impliquant le mappage des structures de données dans un modèle de données aux structures de données équivalentes dans un autre modèle de données. Par exemple, les relations dans le modèle de données relationnelles sont mappées aux enregistrements et aux ensembles dans le modèle de réseau. Il est également nécessaire de traduire le langage de requête utilisé (par exemple, les instructions SQL SELECT sont mappées sur les instructions réseau FIND et GET). Si le matériel et le logiciel sont différents, ces deux types de traduction sont nécessaires. Cela rend le traitement extrêmement complexe.

Une complexité supplémentaire est la fourniture d'un schéma conceptuel commun, qui est formé à partir de l'intégration de schémas conceptuels locaux individuels. Comme vous l'avez déjà vu à l'étape 2.6 de la méthodologie de conception de base de données logique présentée au chapitre 17, l'intégration des modèles de données peut être très difficile en raison de l'hétérogénéité sémantique. Par exemple, des attributs portant le même nom dans deux schémas peuvent représenter des choses différentes. De même, des attributs portant des noms différents peuvent modéliser la même chose. Une discussion complète sur la détection et la résolution de l'hétérogénéité sémantique dépasse le cadre de ce livre. Le lecteur intéressé est renvoyé à l'article de Garcia Solaco et al. (1996).

La solution typique utilisée par certains systèmes relationnels faisant partie d'un SGBDR hétérogène consiste à utiliser des passerelles, qui convertissent le langage et le modèle de chaque SGBD différent dans le langage et le modèle du système relationnel. Cependant, l'approche passerelle présente de sérieuses limitations. Premièrement, il peut ne pas prendre en charge la gestion des transactions, même pour une paire de systèmes ; en d'autres termes, la passerelle entre deux systèmes peut n'être qu'un traducteur de requêtes. Par exemple, un système peut ne pas coordonner le contrôle de la concurrence et la récupération des transactions qui impliquent des mises à jour de la paire de bases de données. Deuxièmement, l'approche passerelle ne concerne que le problème de la traduction d'une requête exprimée dans une langue en une expression équivalente dans une autre langue. En tant que tel, il n'aborde généralement pas les problèmes d'homogénéisation des différences structurelles et de représentation entre les différents schémas.

Accès aux bases de données ouvertes et interopérabilité

L'Open Group a formé un groupe de travail sur les spécifications (SWG) pour répondre à un livre blanc sur l'accès aux bases de données ouvertes et l'interopérabilité (Gualtieri, 1996). Le but de ce groupe était de fournir des spécifications ou de s'assurer que les spécifications

existent ou sont en cours de développement qui créeront un environnement d'infrastructure de base de données où il y aÿ:

- une API SQL commune et puissante qui permet d'écrire des applications clientes qui n'ont pas besoin de connaître le fournisseur du SGBD auquel elles accèdentÿ; un protocole de base de données
- commun qui permet à un SGBD d'un fournisseur de communiquer directement avec un SGBD d'un autre fournisseur sans avoir besoin d'une passerelleÿ; un protocole réseau commun qui permet les
- communications entre différents SGBD.

L'objectif le plus ambitieux est de trouver un moyen de permettre à une transaction de couvrir des bases de données gérées par des SGBD de différents fournisseurs sans utiliser de passerelle. Ce groupe de travail a évolué pour devenir le consortium Database Interoperability (DBIOP), travaillant sur la version 3 de l'architecture de base de données relationnelle distribuée (DRDA), dont nous parlerons brièvement à la section 24.5.2.

Systèmes multibases de données

Avant de terminer cette section, nous présentons brièvement un type particulier de SGBD distribué connu sous le nom de système multibases de données.

Système multibase de données (MDBS) — Un SGBD distribué dans lequel chaque site conserve une autonomie complète.

Ces dernières années, il y a eu un intérêt considérable pour les MDBS, qui tentent d'intégrer logiquement un certain nombre de DDBMS indépendants tout en permettant aux SGBD locaux de garder le contrôle complet de leurs opérations. Une conséquence de l'autonomie complète est qu'il ne peut y avoir aucune modification logicielle des SGBD locaux. Ainsi, un MDBS nécessite une couche logicielle supplémentaire au-dessus des systèmes locaux pour fournir les fonctionnalités nécessaires.

Un MDBS permet aux utilisateurs d'accéder aux données et de les partager sans nécessiter l'intégration complète du schéma de base de données. Cependant, il permet toujours aux utilisateurs d'administrer leurs propres bases de données sans contrôle centralisé, comme avec les vrais DDBMS. Le DBA d'un SGBD local peut autoriser l'accès à des parties particulières de sa base de données en spécifiant un schéma d'exportation, qui définit les parties de la base de données accessibles aux utilisateurs non locaux. Il existe des MDBS non fédérés (où il n'y a pas d'utilisateurs locaux) et fédérés. Un système fédéré est un croisement entre un SGBD distribué et un SGBD centralisé ; il s'agit d'un système distribué pour les utilisateurs mondiaux et d'un système centralisé pour les utilisateurs locaux. La figure 24.4 illustre une taxonomie partielle des SGBD (voir aussi la figure 26.20). Le lecteur intéressé est renvoyé à Sheth et Larson (1990) pour une taxonomie des SGBD distribués et à Bukhres et Elmagarmid (1996).

En termes simples, un MDBS est un SGBD qui réside de manière transparente au-dessus des bases de données et des systèmes de fichiers existants, et présente une base de données unique à ses utilisateurs. Un MDBS ne gère que le schéma global par rapport auquel les utilisateurs émettent des requêtes et des mises à jour et les SGBD locaux eux-mêmes conservent toutes les données utilisateur. Le schéma global est construit en intégrant les schémas des bases de données locales. Le MDBS traduit d'abord les requêtes et mises à jour globales en requêtes et mises à jour sur les SGBD locaux appropriés. Il fusionne ensuite les résultats locaux et génère le résultat global final

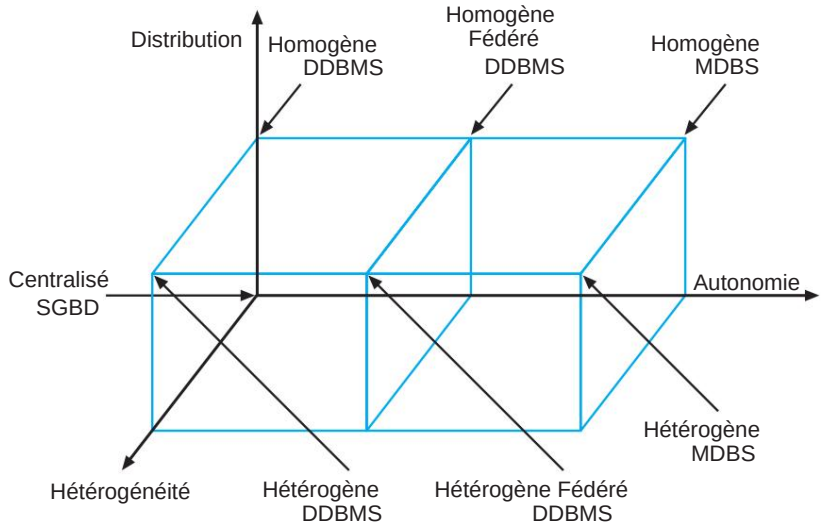

Figure 24.4
Taxonomy of DBMS integration alternatives.

pour l'utilisateur. De plus, le MDBS coordonne les opérations de validation et d'abandon des transactions globales par les SGBD locaux qui les ont traitées, afin de maintenir la cohérence des données dans les bases de données locales. Un MDBS contrôle plusieurs passerelles et gère les bases de données locales via ces passerelles. Nous discutons de l'architecture d'un MDBS dans la section 24.3.3.

24.2 Présentation de la mise en réseau

Réseau Ensemble interconnecté d'ordinateurs autonomes capables d'échanger des informations.

La mise en réseau informatique est un domaine complexe et en évolution rapide, mais une certaine connaissance de celui-ci est utile pour comprendre les systèmes distribués. De la situation d'il y a quelques décennies, lorsque les systèmes étaient autonomes, nous trouvons maintenant les réseaux informatiques monnaie courante. Ils vont de systèmes connectant quelques PC à des réseaux mondiaux avec des milliers de machines et plus d'un million d'utilisateurs. Pour nos besoins, le DDBMS est construit au-dessus d'un réseau de telle manière que le réseau est caché à l'utilisateur.

Les réseaux de communication peuvent être classés de plusieurs manières. Une classification est selon que la distance séparant les ordinateurs est courte (réseau local) ou longue (réseau étendu). Un réseau local (LAN) est destiné à connecter des ordinateurs sur une distance relativement courte, par exemple, dans un immeuble de bureaux, une école ou un collège, ou à la maison. Parfois, un bâtiment contiendra plusieurs petits réseaux locaux et parfois un réseau local s'étendra sur plusieurs bâtiments voisins. Les réseaux locaux sont généralement détenus, contrôlés et gérés par une seule organisation ou personne. Les principales technologies de connectivité sont Ethernet et WiFi. Un réseau étendu (WAN) est utilisé lorsque des ordinateurs ou des réseaux locaux doivent être connectés sur de longues distances. Le plus grand WAN existant est Internet. Contrairement aux réseaux locaux, les réseaux étendus ne sont généralement pas la propriété d'une seule organisation, mais existent plutôt sous une propriété et une gestion collectives ou distribuées. Les WAN utilisent des technologies comme ATM, FrameRelay, SONET/

SDH et X.25 pour la connectivité. Un cas particulier du WAN est un réseau métropolitain (MAN), qui couvre généralement une ville ou une banlieue.

Avec la grande séparation géographique, les liaisons de communication dans un WAN sont relativement lentes et moins fiables que les LAN. Les débits de transmission pour un WAN varient généralement de 33,6 kilobits par seconde (accès commuté via modem) à 45 mégabits par seconde (Mbits/s) (ligne privée non commutée T3) et 274 Mbits/s pour T4. SONET commence d'environ 50 Mbits/s (OC1) à environ 40 Gbits/s (OC768). Les débits de transmission pour les réseaux locaux sont beaucoup plus élevés, fonctionnant de 10 mégabits par seconde (Ethernet partagé) à 2500 Mbits/s (ATM), et sont très fiables. De toute évidence, un DDBMS utilisant un LAN pour la communication fournira un temps de réponse beaucoup plus rapide qu'un utilisant un WAN.

Si nous examinons la méthode de choix d'un chemin ou d'un routage, nous pouvons classer un réseau comme point à point ou comme diffusion. Dans un réseau point à point, si un site souhaite envoyer un message à tous les sites, il doit envoyer plusieurs messages distincts. Dans un réseau de diffusion, tous les sites reçoivent tous les messages, mais chaque message a un préfixe qui identifie le site de destination afin que les autres sites l'ignorent tout simplement. Les WAN sont généralement basés sur un réseau point à point, tandis que les LAN utilisent généralement la diffusion. Un résumé des caractéristiques typiques des WAN et des LAN est présenté dans le Tableau 24.2.

L'Organisation internationale de normalisation a défini un protocole régissant la manière dont les systèmes peuvent communiquer (ISO, 1981). L'approche adoptée consiste à diviser le réseau en une série de couches, chaque couche fournissant un service particulier à la couche supérieure, tout en lui cachant les détails de mise en œuvre. Le protocole, connu sous le nom de modèle d'interconnexion de systèmes ouverts ISO (modèle OSI), se compose de sept couches indépendantes du fabricant. Les couches gèrent la transmission des bits bruts sur le réseau, gèrent la connexion et garantissent que le

TABLEAU 24.2 Summary of typical WAN and LAN characteristics.

VAN	ET
Distances up to thousands of kilometers	Distances up to a few kilometers (Wireless LAN, or WLAN, is of the order of tens of meters)
Link autonomous computers	Link computers that cooperate in distributed applications
Network managed by independent organization (using telephone or satellite links)	Network managed by users (using privately owned cables)
Data rate up to 33.6 kbit/s (dial-up via modem), 45 Mbit/s (T3 circuit)	Data rate up to 2500 Mbit/s (ATM). 100 gigabyte (100 million bits per second) for Ethernet. WLAN is typically 1–108 Mbit/s, although 802.11n standard is 600 Mbits/s.
Complex protocol	Simpler protocol
Use point-to-point routing	Use broadcast routing
Use irregular topology	Use bus or ring topology
Error rate about $1:10^5$	Error rate about $1:10^9$

le lien est exempt d'erreurs, de contrôle du routage et de la congestion, de la gestion des sessions entre différentes machines et de la résolution des différences de format et de représentation des données entre les machines. Une description de ce protocole n'est pas nécessaire pour comprendre ces trois chapitres sur les SGBD distribués et mobiles et nous renvoyons donc le lecteur intéressé à Halsall (1995) et Tanenbaum (1996).

Le Comité consultatif international télégraphique et téléphonique (CCITT) a produit une norme connue sous le nom de X.25 qui est conforme aux trois couches inférieures de cette architecture. La plupart des DDBMS ont été développés sur X.25. Cependant, de nouvelles normes sont produites pour les couches supérieures qui peuvent fournir des services utiles pour les DDBMS, par exemple, Remote Database Access (RDA) (ISO 9579) ou Distributed Transaction Processing (DTP) (ISO 10026). Nous examinons la norme X/Open DTP à la section 25.5. Comme informations de base supplémentaires, nous fournissons maintenant un bref aperçu des principaux protocoles de mise en réseau.

Protocoles réseau

Protocole réseau	Ensemble de règles qui déterminent comment les messages entre ordinateurs sont envoyés, interprétés et traités.

Dans cette section, nous décrivons brièvement les principaux protocoles réseau.

TCP/IP (Transmission Control Protocol/Internet Protocol) Il s'agit du protocole de communication standard pour Internet, un ensemble mondial de réseaux informatiques interconnectés. TCP est chargé de vérifier la livraison correcte des données du client au serveur. IP fournit le mécanisme de routage, basé sur une adresse de destination de quatre octets (l'adresse IP). La partie avant de l'adresse IP indique la partie réseau de l'adresse et la partie arrière indique la partie hôte de l'adresse. La ligne de démarcation entre les parties réseau et hôte d'une adresse IP n'est pas fixe. TCP/IP est un protocole routable, ce qui signifie que tous les messages contiennent non seulement l'adresse de la station de destination, mais également l'adresse d'un réseau de destination. Cela permet aux messages TCP/IP d'être envoyés à plusieurs réseaux au sein d'une organisation ou dans le monde entier, d'où son utilisation sur Internet.

SPX/IPX (Échange de paquets séquencé/Échange de paquets Internet)
Novell a créé SPX/IPX dans le cadre de son système d'exploitation NetWare. Semblable à TCP, SPX garantit qu'un message entier arrive intact mais utilise le protocole IPX de NetWare comme mécanisme de livraison. Comme IP, IPX gère le routage des paquets sur le réseau. Contrairement à IP, IPX utilise un espace d'adressage de 80 bits, avec une partie réseau de 32 bits et une partie hôte de 48 bits (c'est beaucoup plus grand que l'adresse 32 bits utilisée par IP). De plus, contrairement à IP, IPX ne gère pas la fragmentation des paquets. Cependant, l'une des grandes forces d'IPX est son adressage automatique des hôtes. Les utilisateurs peuvent déplacer leur PC d'un endroit du réseau à un autre et reprendre le travail simplement en le branchant. Ceci est particulièrement important pour les utilisateurs mobiles. Jusqu'à NetWare 5, SPX/IPX était le protocole par défaut, mais pour refléter l'importance d'Internet, NetWare 5 a adopté TCP/IP comme protocole par défaut.

NetBIOS (Network Basic Input/Output System) Un protocole réseau développé en 1984 par IBM et Sytek comme standard pour les communications des applications PC.

À l'origine, NetBIOS et NetBEUI (NetBIOS Extended User Interface) étaient considérés comme un seul protocole. Plus tard, NetBIOS a été supprimé car il pouvait être utilisé avec d'autres protocoles de transport routables, et maintenant les sessions NetBIOS peuvent être transportées via les protocoles NetBEUI, TCP/IP et SPX/IPX. NetBEUI est un protocole petit, rapide et effi cace. Cependant, il n'est pas routable, donc une configuration typique utilise NetBEUI pour la communication avec un LAN et TCP/IP au-delà du LAN.

APPC (Advanced ProgramtoProgram Communications) Un protocole de communication de haut niveau d'IBM qui permet à un programme d'interagir avec un autre à travers le réseau. Il prend en charge le client-serveur et l'informatique distribuée en fournissant une interface de programmation commune à toutes les plates-formes IBM. Il offre commandes pour gérer une session, envoyer et recevoir des données, et transaction gestion à l'aide d'un commit en deux phases (dont nous parlerons dans le chapitre suivant). Le logiciel APPC fait partie ou est disponible en option sur tous les systèmes d'exploitation IBM et de nombreux systèmes d'exploitation non IBM. Étant donné qu'APPC ne prenait en charge à l'origine que les systèmes IBM Architecture réseau, qui utilise le protocole LU 6.2 pour l'établissement de session, APPC et LU 6.2 sont parfois considérés comme synonymes.

DECnet DECnet est le protocole de communication routable de Digital, qui prend en charge Réseaux locaux de type Ethernet et réseaux étendus à bande de base et à large bande sur des réseaux privés ou publics lignes. Il interconnecte les PDP, les VAX, les PC, les Macintosh et les postes de travail.

AppleTalk Il s'agit du protocole LAN routable d'Apple, introduit en 1985, qui prend en charge la méthode d'accès propriétaire LocalTalk d'Apple ainsi que l'Ethernet et le jeton. anneau. Le gestionnaire de réseau AppleTalk et la méthode d'accès LocalTalk sont construits dans tous les Macintosh et LaserWriters. Il a depuis été déprécié par Apple en faveur de TCP/IP.

WAP (Wireless Application Protocol) Une norme pour fournir des téléphones, téléavertisseurs et autres appareils portables avec un accès sécurisé aux pages Web basées sur le courrier électronique et le texte. Introduit en 1997 par Phone.com (anciennement Unwired Planet), Ericsson, Motorola et Nokia, WAP fournit un environnement complet pour le sans fil applications qui incluent une contrepartie sans fil de TCP/IP et un cadre pour l'intégration de la téléphonie comme le contrôle des appels et l'accès au répertoire téléphonique.

Temps de communication

Le temps nécessaire pour envoyer un message dépend de la longueur du message et le type de réseau utilisé. Il peut être calculé à l'aide de la formule suivante :

Temps de communication 5 C0 1 (no_of_bits_in_message/transmission_rate)

où C0 est un coût fixe d'initiation d'un message, appelé délai d'accès. Par exemple, en utilisant un délai d'accès de 1 seconde et un taux de transmission de 10 000 bits par seconde, nous pouvons calculer le temps d'envoi de 100 000 enregistrements, chacun composé de 100 bits comme suitÿ:

Temps de communication 5 1 1 (100 000*100/10 000) 5 1001 secondes

Si nous souhaitons transférer 100 000 enregistrements un par un, nous obtenons :

Temps de communication 5 100 000 * [1 1 (100/10 000)]
5 100 000 * [1.01] 5 101 000 secondes

En clair, le temps de communication pour transférer 100 000 enregistrements individuellement est significativement plus long, à cause du délai d'accès. Par conséquent, un objectif d'un DDBMS est de minimiser à la fois le volume de données transmises sur le réseau et le nombre de transmissions réseau.

Nous revenons sur ce point lorsque nous considérons l'optimisation distribuée des requêtes dans la section 24.5.3.

24.3 Fonctions et architectures d'un DDBMS

Au chapitre 2, nous avons examiné les fonctions, l'architecture et les composants d'un SGBD centralisé. Dans cette section, nous examinons comment la distribution affecte la fonctionnalité et l'architecture attendues.

24.3.1 Fonctions d'un DDBMS

Nous nous attendons à ce qu'un DDBMS ait au moins les fonctionnalités d'un SGBD centralisé dont nous avons parlé au chapitre 2. De plus, nous nous attendons à ce qu'un DDBMS ait les fonctionnalités suivantesÿ:

- services de communication étendus pour fournir un accès à des sites distants et permettre le transfert de requêtes et de données entre les sites à l'aide d'un réseauÿ; catalogue système étendu pour stocker
- les détails de distribution des donnéesÿ; le traitement distribué des requêtes, y compris l'optimisation
- des requêtes et l'accès aux données à distanceÿ; un contrôle de sécurité étendu pour maintenir les
- privilèges d'autorisation/d'accès appropriés aux données distribuéesÿ; contrôle de concurrence étendu pour maintenir la cohérence des données distribuées et éventuellement répliquéesÿ; des services de
- récupération étendus pour tenir compte des défaillances des sites individuels et des défaillances des liaisons de communication.
-

Nous abordons ces questions plus en détail dans les sections ultérieures de ce chapitre et au chapitre 25.

24.3.2 Architecture de référence pour un DDBMS

L'architecture à trois niveaux ANSISPARC pour un SGBD présentée dans la section 2.1 fournit une architecture de référence pour un SGBD centralisé. Du fait de la diversité des SGBD distribués, il est beaucoup plus difficile de présenter une architecture équivalente généralisable. Cependant, il peut être utile de présenter une architecture de référence possible qui traite de la distribution des données. L'architecture de référence illustrée à la Figure 24.5 se compose des schémas suivantsÿ:

- un ensemble de schémas externes
- globauxÿ; un schéma conceptuel global ;
- un schéma de fragmentation et un schéma d'allocationÿ; un
- ensemble de schémas pour chaque SGBD local conforme à l'architecture à trois niveaux ANSISPARC.

Les bords de cette figure représentent les mappages entre les différents schémas. Selon les niveaux de transparence pris en charge, certains niveaux peuvent manquer dans l'architecture.

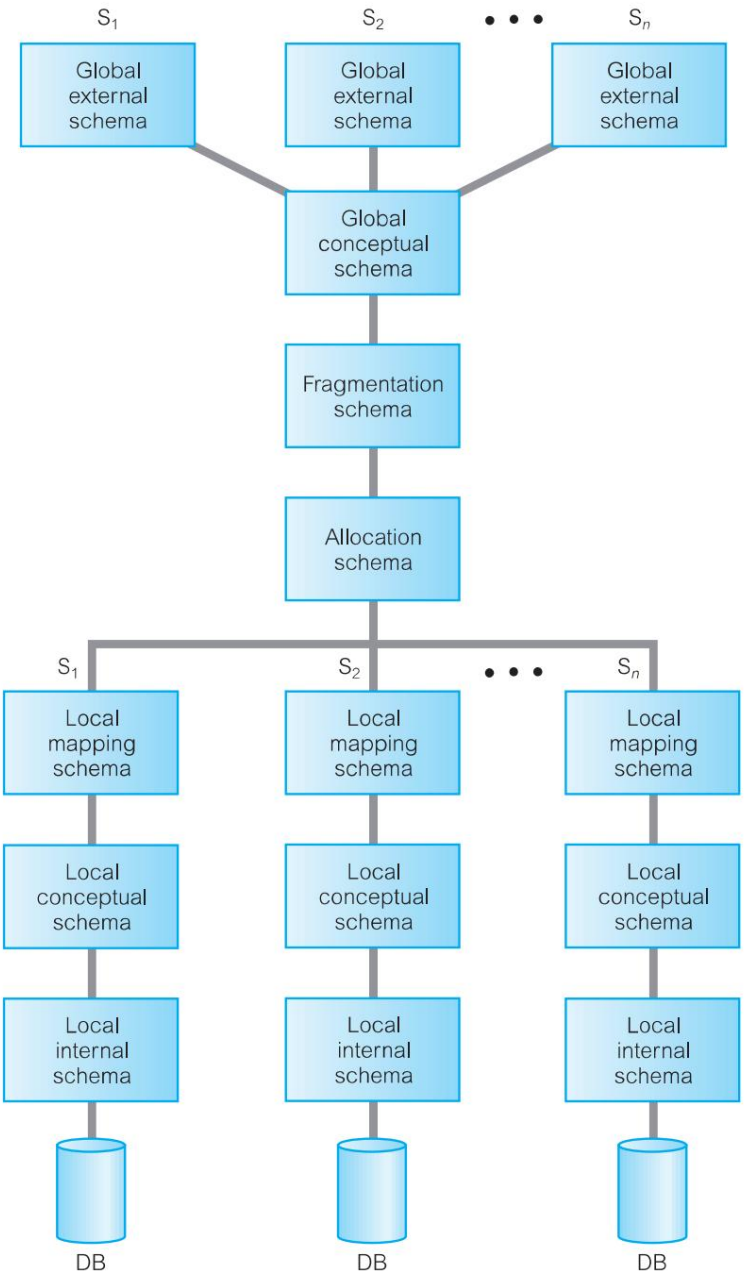

Figure 24.5
Reference architecture for a DDBMS.

Schéma conceptuel global Le

schéma conceptuel global est une description logique de l'ensemble de la base de données, comme si elle n'était pas distribuée. Ce niveau correspond au niveau conceptuel de l'architecture ANSI SPARC et contient des définitions d'entités, de relations, de contraintes, d'informations de sécurité et d'intégrité. Il offre une indépendance des données physiques par rapport à l'environnement distribué. Les schémas externes globaux offrent une indépendance logique des données.

Schémas de fragmentation et d'allocation

Le schéma de fragmentation est une description de la façon dont les données doivent être logiquement partitionnées. Le schéma d'allocation est une description de l'endroit où les données doivent être localisées, en tenant compte de toute réplication.

Schémas locaux

Chaque SGBD local possède son propre ensemble de schémas. Les schémas conceptuels locaux et internes locaux correspondent aux niveaux équivalents de l'architecture ANSISPARC.
Le schéma de mappage local mappe des fragments du schéma d'allocation dans des objets dans la base de données locale. Il est indépendant du SGBD et constitue la base du support des SGBD hétérogènes.

24.3.3 Architecture de référence pour un MDBS fédéré

Dans la section 24.1.3, nous avons brièvement discuté des systèmes de bases de données multiples fédérées (FMDBS). Les systèmes fédérés diffèrent des DDBMS par le niveau d'autonomie locale fourni.
Cette différence se reflète également dans l'architecture de référence. La figure 24.6 illustre une architecture de référence pour un FMDBS étroitement couplé, c'est-à-dire qu'il a

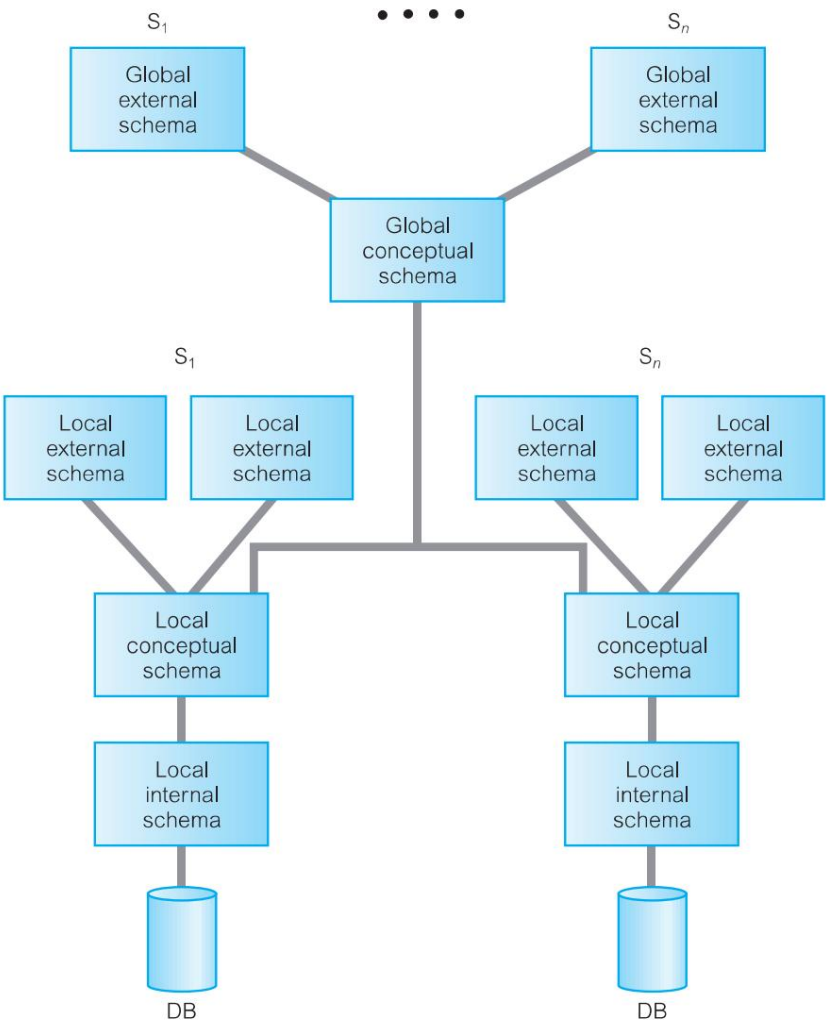

Figure 24.6
Reference architecture for a tightly coupled FMDBS.

un schéma conceptuel global (GCS). Dans un DDBMS, le GCS est l'union de tous les schémas conceptuels locaux. Dans un FMDBS, le GCS est un sous-ensemble des schémas conceptuels locaux, composé des données que chaque système local accepte de partager. Le GCS d'un système étroitement couplé implique l'intégration soit des parties des schémas conceptuels locaux, soit des schémas externes locaux.

Il a été avancé qu'un FMDBS ne devrait pas avoir de GCS (Litwin, 1988), auquel cas le système est qualifié de faiblement couplé. Dans ce cas, les schémas externes consistent en un ou plusieurs schémas conceptuels locaux. Pour plus d'informations sur les MDBS, le lecteur intéressé est renvoyé à Litwin (1988) et Sheth et Larson (1990).

24.3.4 Architecture de composants pour un DDBMS Indépendamment

de l'architecture de référence, nous pouvons identifier une architecture de composants pour un DDBMS constituée de quatre composants principauxÿ:

- composant SGBD local (LDBMS)ÿ;
- composant de communication de données
- (DC); catalogue global du système (GSC);
- composant de SGBD distribué (DDBMS).

L'architecture des composants d'un DDBMS basé sur la Figure 24.1 est illustrée à la Figure 24.7. Pour plus de clarté, nous avons omis le site 2 du diagramme, car il a le même structure en tant que Site 1.

Composant SGBD local Le

composant LDBMS est un SGBD standard, responsable du contrôle des données locales sur chaque site disposant d'une base de données. Il possède son propre catalogue de système local qui stocke des informations sur les données détenues sur ce site. Dans un système homogène, le LDBMS

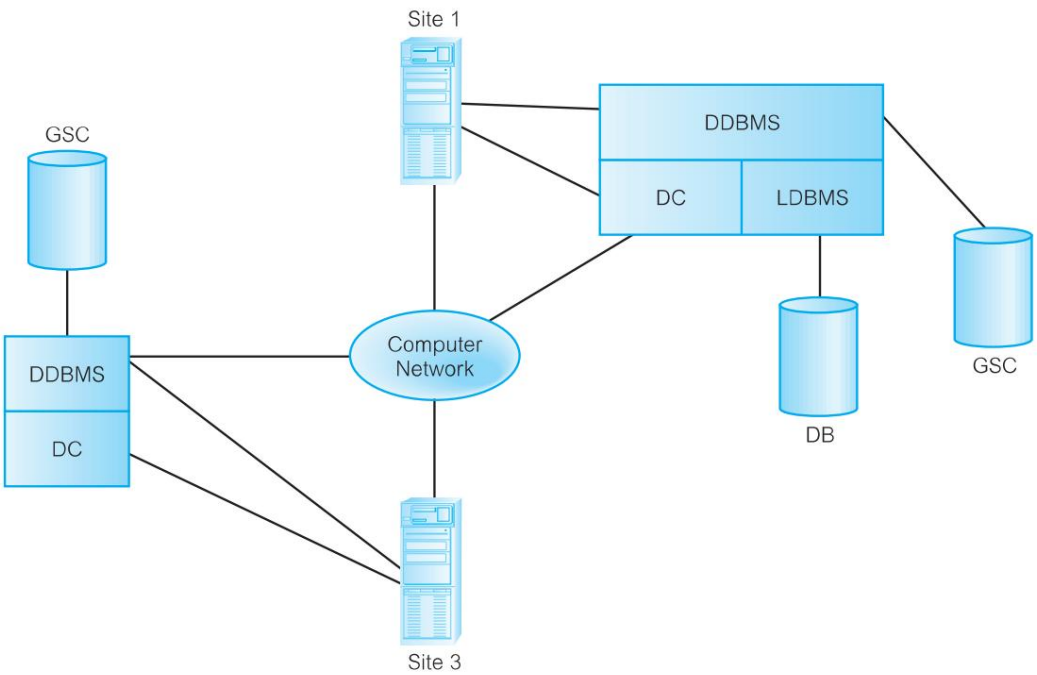

Figure 24.7
Components of a DDBMS.

composant est le même produit, reproduit sur chaque site. Dans un système hétérogène, il y aurait au moins deux sites avec différents produits et/ou plates-formes SGBD.

Composant de communication de données

Le composant DC est le logiciel qui permet à tous les sites de communiquer entre eux. Le composant DC contient des informations sur les sites et les liens.

Catalogue système global

Le GSC a les mêmes fonctionnalités que le catalogue système d'un système centralisé. Le GSC contient des informations spécifiques à la nature distribuée du système, telles que les schémas de fragmentation, de réplication et d'allocation. Elle peut elle-même être gérée comme une base de données distribuée et peut donc être fragmentée et distribuée, entièrement répliquée ou centralisée, comme toute autre relation, comme nous le verrons bientôt. Un GSC entièrement répliqué compromet l'autonomie du site, car chaque modification du GSC doit être communiquée à tous les autres sites. Un GSC centralisé compromet également l'autonomie du site et est vulnérable aux défaillances du site central.

L'approche adoptée dans le système distribué R* pallie ces défauts (Williams et al., 1982). Dans R*, il existe un catalogue local sur chaque site qui contient les métadonnées relatives aux données stockées sur ce site. Pour les relations créées sur un site (le site de naissance), il est de la responsabilité du catalogue local de ce site d'enregistrer la définition de chaque fragment et de chaque réplique de chaque fragment, et d'enregistrer l'emplacement de chaque fragment ou réplique. Chaque fois qu'un fragment ou une réplique est déplacé vers un emplacement différent, le catalogue local du site de naissance de la relation correspondante doit être mis à jour. Ainsi, pour localiser un fragment ou une réplique d'une relation, il faut accéder au catalogue du lieu de naissance de la relation. Le site de naissance de chaque relation globale est enregistré dans chaque GSC local. Nous revenons à la dénomination d'objet lorsque nous discutons de la transparence de la dénomination dans la section 24.5.1.

Composant SGBD distribué Le

composant DDBMS est l'unité de contrôle de l'ensemble du système. Nous avons brièvement énuméré la fonctionnalité de ce composant dans la section précédente et nous nous concentrons sur cette fonctionnalité à la section 24.5 et au chapitre 25.

24.4 Conception de bases de données relationnelles distribuées

Dans les chapitres 16 et 17, nous avons présenté une méthodologie pour la conception conceptuelle et logique d'une base de données relationnelle centralisée. Dans cette section, nous examinons les facteurs supplémentaires qui doivent être pris en compte pour la conception d'une base de données relationnelle distribuée. Plus précisément, nous examinons :

- Fragmentation. Une relation peut être divisée en un certain nombre de sous-relations, appelées fragments, qui sont ensuite distribuées. Il existe deux grands types de fragmentation : horizontale et verticale. Les fragments horizontaux sont des sous-ensembles de tuples et les fragments verticaux sont des sous-ensembles d'attributs.
- Allocation. Chaque fragment est stocké sur le site avec une distribution « optimale ».
- Réplication. Le DDBMS peut conserver une copie d'un fragment sur plusieurs sites différents.

La définition et l'allocation des fragments doivent être basées sur la façon dont la base de données doit être utilisée. Il s'agit d'analyser les transactions. Généralement, il n'est pas possible d'analyser toutes les transactions, nous nous concentrons donc sur les plus importantes. Comme indiqué dans la section 18.2, il a été suggéré que les 20ÿ% les plus actifs des requêtes des utilisateurs représentent 80ÿ% de l'accès total aux données, et cette règle 80/20 peut être utilisée comme ligne directrice dans la réalisation de l'analyse (Wiederhold, 1983).

La conception doit être basée sur des informations quantitatives et qualitatives. Des informations quantitatives sont utilisées dans l'allocation ; des informations qualitatives sont utilisées dans la fragmentation. Les informations quantitatives peuvent inclureÿ:

- la fréquence à laquelle une transaction est exécutéeÿ; le
- site à partir duquel une transaction est exécutéeÿ; les
- critères de performance des transactions.

Les informations qualitatives peuvent inclure des informations sur les transactions qui sont exécutées, telles queÿ:

- les relations, attributs et tuples auxquels on accèdeÿ; le
- type d'accès (lecture ou écriture) ; les prédicats des
- opérations de lecture.

La définition et l'attribution des fragments sont effectuées de manière stratégique pour atteindre les objectifs suivantsÿ:

- Localité de référence. Dans la mesure du possible, les données doivent être stockées à proximité de l'endroit où elles sont utilisées. Si un fragment est utilisé sur plusieurs sites, il peut être avantageux de stocker des copies du fragment sur ces sites.
- Fiabilité et disponibilité améliorées. La fiabilité et la disponibilité sont améliorées par la réplication : une autre copie du fragment est disponible sur un autre site en cas de défaillance d'un site.
- Performances acceptables. Une mauvaise allocation peut entraîner l'apparition de goulots d'étranglementÿ; c'est-à-dire qu'un site peut être inondé de demandes provenant d'autres sites, ce qui peut entraîner une dégradation significative des performances. Alternativement, une mauvaise allocation peut entraîner une sous-utilisation des ressources.
- Capacités de stockage et coûts équilibrés. Il convient de tenir compte de la disponibilité et du coût du stockage sur chaque site, de sorte qu'un stockage de masse bon marché puisse être utilisé dans la mesure du possible. Cela doit être mis en balance avec la localité de référence.
- Coûts de communication minimes. Il convient de tenir compte du coût des demandes à distance. Les coûts de récupération sont minimisés lorsque la localité de référence est maximisée ou lorsque chaque site possède sa propre copie des données. Cependant, lorsque les données répliquées sont mises à jour, la mise à jour doit être effectuée sur tous les sites détenant une copie en double, ce qui augmente les coûts de communication.

24.4.1 Attribution des données

Il existe quatre stratégies alternatives concernant le placement des donnéesÿ: centralisée, fragmentée, réplication complète et réplication sélective. Nous comparons maintenant ces stratégies en utilisant les objectifs identifiés précédemment.

Centralisé

Cette stratégie consiste en une seule base de données et un SGBD stockés sur un site avec des utilisateurs répartis sur le réseau (nous l'avons précédemment appelé traitement distribué). La localité de référence est au plus bas car tous les sites, à l'exception du site central, doivent utiliser le réseau pour tous les accès aux données. Cela signifie également que les coûts de communication sont élevés. La fiabilité et la disponibilité sont faibles, car une défaillance du site central entraîne la perte de l'ensemble du système de base de données.

Fragmenté (ou partitionné)

Cette stratégie partitionne la base de données en fragments disjoints, chaque fragment étant affecté à un site. Si les éléments de données sont situés sur le site où ils sont le plus fréquemment utilisés, la localité de référence est élevée. Comme il n'y a pas de réplication, les coûts de stockage sont faibles ; de même, la fiabilité et la disponibilité sont faibles, bien qu'elles soient plus élevées que dans le cas centralisé, car la défaillance d'un site entraîne la perte des seules données de ce site. Les performances doivent être bonnes et les coûts de communication faibles si la distribution est correctement conçue.

Réplication complète Cette

stratégie consiste à maintenir une copie complète de la base de données sur chaque site. Par conséquent, la localité de référence, la fiabilité, la disponibilité et les performances sont maximisées. Cependant, les coûts de stockage et les coûts de communication pour les mises à jour sont les plus onéreux. Pour surmonter certains de ces problèmes, des instantanés sont parfois utilisés. Un instantané est une copie des données à un instant donné. Les copies sont mises à jour périodiquement, par exemple toutes les heures ou toutes les semaines, de sorte qu'elles ne sont pas toujours à jour. Les instantanés sont également parfois utilisés pour implémenter des vues dans une base de données distribuée afin d'améliorer le temps nécessaire pour effectuer une opération de base de données sur une vue. Nous discutons des instantanés dans la section 26.3.

Réplication sélective Cette

stratégie est une combinaison de fragmentation, de réplication et de centralisation. Certains éléments de données sont fragmentés pour obtenir une localité de référence élevée, et d'autres qui sont utilisés sur de nombreux sites et ne sont pas fréquemment mis à jour sont répliquésÿ; dans le cas contraire, les éléments de données sont centralisés. L'objectif de cette stratégie est d'avoir tous les avantages des autres approches mais aucun des inconvénients. C'est la stratégie la plus couramment utilisée, en raison de sa flexibilité. Les stratégies alternatives sont résumées dans le tableau 24.3. Pour plus de détails sur l'allocation, le lecteur intéressé est renvoyé à Ozsu et Valduriez (1999) et Teorey (1994).

24.4.2 Fragmentation

Pourquoi fragmenter ?

Avant de discuter de la fragmentation en détail, nous énumérons quatre raisons de fragmenter une relationÿ:

- Usage. En général, les applications fonctionnent avec des vues plutôt qu'avec des relations entières. Par conséquent, pour la distribution des données, il semble approprié de travailler avec des sous-ensembles de relations comme unité de distribution.

TABLEAU 24.3 Comparison of strategies for data allocation.

	LOCALITÉ DE RÉFÉRENCE	FIABILITÉ ET DISPONIBILITÉ	PERFORMANCE	STOCKAGE FRAIS	LA COMMUNICATION FRAIS
Centralized	Lowest	Lowest	Unsatisfactory	Lowest	Highest
Fragmented	High[a]	Low for item; high for system	Satisfactory[a]	Lowest	Low[a]
Complete replication	Highest	Highest	Best for read	Highest	High for update; low for read
Selective replication	High[a]	Low for item; high for system	Satisfactory[a]	Average	Low[a]

[a]Indicates subject to good design.

- Efficacité. Les données sont stockées à proximité de l'endroit où elles sont le plus fréquemment utilisées. De plus, les données qui ne sont pas nécessaires aux applications locales ne sont pas stockées.
- Parallélisme. Avec des fragments comme unité de distribution, une transaction peut être divisée en plusieurs sous-requêtes qui fonctionnent sur des fragments. Cela devrait augmenter le degré de concurrence, ou de parallélisme, dans le système, permettant ainsi aux transactions qui peuvent le faire en toute sécurité de s'exécuter en parallèle.
- Sécurité. Les données non requises par les applications locales ne sont pas stockées et ne sont donc pas accessibles aux utilisateurs non autorisés.

La fragmentation présente deux principaux inconvénients, que nous avons mentionnés précédemmentÿ:

- Performance. Les performances des applications globales qui nécessitent des données provenant de plusieurs fragments situés sur différents sites peuvent être plus lentes.
- Intégrité. Le contrôle de l'intégrité peut être plus difficile si les données et les dépendances fonctionnelles sont fragmentées et situées sur des sites différents.

Exactitude de la fragmentation

La fragmentation ne peut pas être effectuée au hasard. Trois règles doivent être suivies lors de la fragmentationÿ:

(1) Intégralité. Si une instance de relation est décomposée R en fragments 1, 2,..., , chaque donnée pouvant être trouvée R_n dans doit apparaître dans au moins un fragment. Cette règle est nécessaire pour s'assurer qu'il n'y a pas de perte de données pendant la fragmentation.

(2) Reconstruction. Il doit être possible de définir une opération relationnelle qui reconstruira la relation à partir des fragments. Cette règle garantit que R les dépendances fonctionnelles sont préservées.

(3) Disjonction. Si un élément de données i , d apparaît dans fragment , il ne doit R apparaître dans aucun autre fragment. La fragmentation verticale est l'exception à cette règle, où les attributs de clé primaire doivent être répétés pour permettre la reconstruction. Cette règle garantit une redondance minimale des données.

Dans le cas de la fragmentation horizontale, une donnée est un tuple ; pour la fragmentation verticale, un élément de données est un attribut.

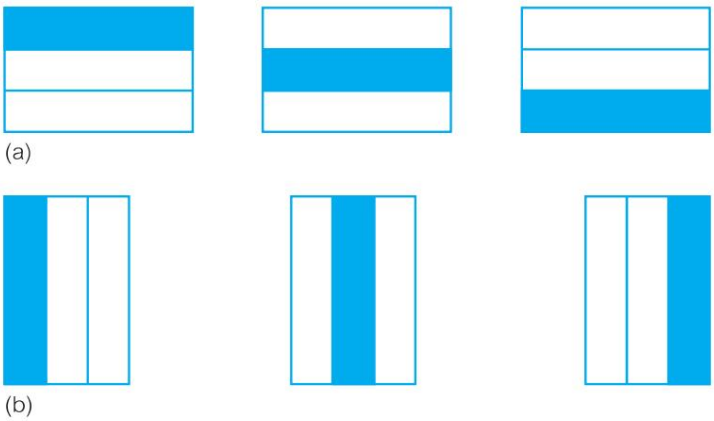

Figure 24.8 (a) Horizontal and (b) vertical fragmentation.

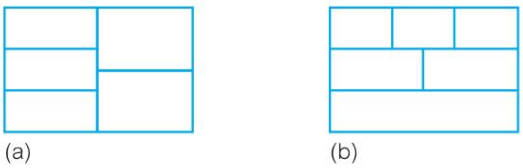

Figure 24.9 Mixed fragmentation: (a) vertical fragments, horizontally fragmented; (b) horizontal fragments, vertically fragmented.

Types de fragmentation II

existe deux principaux types de fragmentation : horizontale et verticale. Les fragments horizontaux sont des sous-ensembles de tuples et les fragments verticaux sont des sous-ensembles d'attributs, comme illustré à la figure 24.8. Il existe également deux autres types de fragmentation : mixte, illustrée à la figure 24.9, et dérivée, un type de fragmentation horizontale. Nous fournissons maintenant des exemples des différents types de fragmentation en utilisant l'instance de la base de données DreamHome illustrée à la figure 4.3.

Fragmentation horizontale

Fragment horizontal	Se compose d'un sous-ensemble des tuples d'une relation.

La fragmentation horizontale regroupe les tuples dans une relation qui sont collectivement utilisés par les transactions importantes. Un fragment horizontal est produit en spécifiant un prédicat qui effectue une restriction sur les tuples dans la relation. Il est défini à l'aide de l'opération Sélection de l'algèbre relationnelle (voir Section 5.1.1). L'opération Selection regroupe les tuples qui ont une propriété commune ; par exemple, les tuples sont tous utilisés par la même application ou sur le même site. Étant donné une relation , un fragment horizontal est défini comme : R

$$\sigma_p(R)$$

où p est un prédicat basé sur un ou plusieurs attributs de la relation.

EXEMPLE 24.2 Fragmentation horizontale

En supposant qu'il n'y ait que deux types de biens, Appartement et Maison, la fragmentation horizontale de PropertyForRent par type de bien peut être obtenue comme suitÿ:

P1 : type 5 'Maison' (PropertyForRent)
P2 : type 5 'Appartement' (PropertyForRent)

Cela produit deux fragments (P1 et P2), l'un composé des tuples où la valeur de l'attribut type est 'House' et l'autre composé des tuples où la valeur de l'attribut type est 'Flat', comme le montre la Figure 24.10 . Cette stratégie de fragmentation particulière peut être avantageuse s'il existe des demandes distinctes traitant de maisons et d'appartements (également appelés appartements). Le schéma de fragmentation satisfait les règles de correctionÿ:

Fragment P_1

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003

Fragment P_2

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	SG14	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Image 24.10 Horizontal fragmentation of PropertyForRent by property type.

- Complétude. Chaque tuple de la relation apparaît dans le fragment P1 ou P2.
- Reconstruction. La relation PropertyForRent peut être reconstruite à partir des fragments à l'aide de l'opération Unionÿ:

  P1 ø P2 5 PropertyForRent

- Disjonction. Les fragments sont disjoints ; il ne peut y avoir aucun type de propriété qui soit à la fois 'Maison' et 'Appartement'.

Parfois, le choix de la stratégie de fragmentation horizontale est évident. Cependant, dans d'autres cas, il est nécessaire d'analyser les applications en détail. L'analyse consiste à examiner les prédicats (ou conditions de recherche) utilisés par les transactions ou les requêtes dans les applications. Les prédicats peuvent être simples, impliquant des attributs uniques, ou complexes, impliquant plusieurs attributs. Les prédicats de chaque attribut peuvent être à valeur unique ou à valeurs multiples. Dans ce dernier cas, les valeurs peuvent être discrètes ou impliquer des plages de valeurs.

La stratégie de fragmentation consiste à trouver un ensemble de prédicats minimaux (c'est-à-dire complets et pertinents) pouvant servir de base au schéma de fragmentation (Ceri et al., 1982). Un ensemble de prédicats est complet si et seulement si deux tuples du même fragment sont référencés avec la même probabilité par une transaction. Un prédicat est pertinent s'il existe au moins une transaction qui accède différemment aux fragments résultants. Par exemple, si la seule exigence est de sélectionner des tuples dans PropertyForRent en fonction du type de propriété, l'ensemble {type 5 'House', type 5 'Flat'}

est complet, alors que l'ensemble {type 5 'Maison'} n'est pas complet. D'autre part, avec cette exigence, le prédicat (ville 5 'Aberdeen') ne serait pas pertinent.

Fragmentation verticale

Fragment vertical	Se compose d'un sous-ensemble des attributs d'une relation.

La fragmentation verticale regroupe les attributs dans une relation qui sont utilisés conjointement par les transactions importantes. Un fragment vertical est défini à l'aide de l'opération Projection de l'algèbre relationnelle (voir Section 5.1.1). Étant donné une relation , un fragment vertical R est défini comme :

$$P_{a1, \ldots, a_n}(R)$$

où a1, . . . , a sont des attributs de la relation . $_n$ R

EXEMPLE 24.3 Fragmentation verticale

L'application de paie DreamHome nécessite le numéro de personnel staffNo et le poste, le sexe, la date de naissance et les attributs de salaire de chaque membre du personnelÿ; le service RH a besoin des attributs staffNo, fName, lName et branchNo. La fragmentation verticale de Staff pour cet exemple peut être obtenue comme suit :

S1ÿ: PstaffNo, poste, sexe, date de naissance, salaire (personnel)

S2ÿ: PstaffNo, fName, lName, branchNo(Personnel)

Cela produit deux fragments (S1 et S2), comme le montre la figure 24.11. Notez que les deux fragments contiennent la clé primaire, staffNo, pour permettre la reconstruction de la relation d'origine. L'avantage de la fragmentation verticale est que les fragments peuvent être stockés sur les sites qui en ont besoin. De plus, les performances sont améliorées car le fragment est plus petit que la relation de base d'origine. Ce schéma de fragmentation satisfait les règles de correctionÿ:

Fragment S_1

staffNo	position	sex	DOB	salary
SL21	Manager	M	1-Oct-45	30000
SG37	Assistant	F	10-Nov-60	12000
SG14	Supervisor	M	24-Mar-58	18000
SA9	Assistant	F	19-Feb-70	9000
SG5	Manager	F	3-Jun-40	24000
SL41	Assistant	F	13-Jun-65	9000

Fragment S_2

staffNo	fName	lName	branchNo
SL21	John	White	B005
SG37	Ann	Beech	B003
SG14	David	Ford	B003
SA9	Mary	Howe	B007
SG5	Susan	Brand	B003
SL41	Julie	Lee	B005

Figure 24.11 Vertical fragmentation of Staff.

- Complétude. Chaque attribut de la relation Staff apparaît dans le fragment S1 ou S2.
- Reconstruction. La relation Staff peut être reconstruite à partir des fragments en utilisant la Opération de jointure naturelleÿ:

  S1 1 S2 5 Personnel

- Disjonction. Les fragments sont disjoints à l'exception de la clé primaire, qui est nécessaire pour la reconstruction.

Les fragments verticaux sont déterminés en établissant l'affinité d'un attribut avec une autre. Une façon de faire est de créer une matrice qui montre le nombre d'accès qui font référence à chaque paire d'attributs. Par exemple, une transaction qui accède aux attributs *a*1, *a*2 et 4 de *a*relation avec les *R*attributs (1, 2, 3, 4), peuvent être *a*représentés par le matrice suivanteÿ:

	un_1	un_2	un_3	un_4
un_1		1	0	1
un_2			0	1
un_3				0
un_4				

La matrice est triangulaire ; la diagonale n'a pas besoin d'être remplie car la partie inférieure la moitié est une image miroir de la moitié supérieure. Les 1 représentent un accès impliquant le paire d'attributs correspondante, et sont éventuellement remplacés par des nombres représentant la fréquence des transactions. Une matrice est produite pour chaque transaction et un bilan global Une matrice est produite montrant la somme de tous les accès pour chaque paire d'attributs. Paires avec une haute affinité doit apparaître dans le même fragment verticalÿ; les paires de faible affinité peuvent être séparées. De toute évidence, travailler avec des attributs uniques et toutes les transactions majeures peut être un long calcul. Par conséquent, si l'on sait que certains attributs sont liés, il peut être prudent de travailler avec des groupes d'attributs à la place.

Cette approche est connue sous le nom de fractionnement et a été proposée pour la première fois par Navathe et al. (1984). Il produit un ensemble de fragments non superposés, ce qui garantit la conformité avec la règle de disjonction définie précédemment. En fait, la caractéristique de non-chevauchement s'applique uniquement aux attributs qui ne font pas partie de la clé primaire. Champs de clé primaire apparaissent dans chaque fragment et peuvent donc être omis de l'analyse. Pour plus informations sur cette approche, le lecteur est renvoyé à Ozsu et Valduriez (1999).

Fragmentation mixte Pour certaines applications, la fragmentation horizontale ou verticale d'un schéma de base de données est en elle-même insuffisante pour répartir correctement les données. Au lieu de cela, une fragmentation mixte ou hybride est requise.

Fragment mixte	Se compose d'un fragment horizontal qui est ensuite fragmenté verticalement ou d'un fragment vertical qui est ensuite fragmenté horizontalement.

Un fragment mixte est défini à l'aide des opérations de sélection et de projection du algèbre relationnelle. Étant donné une relation *R*, un fragment mixte est défini comme :

$$sp(Pa_1, \ldots, un \quad (R))$$

ou alors

Hé bien
l , . . . , un (sp( ))R

où p est un prédicat basé sur un ou plusieurs attributs de et de . R $a_{1, . . .,}$ $a_n$sont des attributs
R

EXEMPLE 24.4 Fragmentation mixte

Dans l'exemple 24.3, nous avons fragmenté verticalement le personnel des services de paie et des ressources humaines enÿ:

S1ÿ: PstaffNo, poste, sexe, date de naissance, salaire (personnel)

S2 : PstaffNo, fName, lName, branchNo(Personnel)

Nous pourrions maintenant fragmenter horizontalement S2 en fonction du numéro de branche (pour simplifier, nous supposons qu'il n'y a que trois branches) :

S21ÿ: branche n° 5 'B003' (S2)

S23ÿ: branche n° 5 ' B005 ' (S2)

S23ÿ: branche n° 5 'B007' (S2)

Cela produit trois fragments (S21, S22 et S23), l'un composé de ces tuples où le numéro de branche est B003 (S21), l'un composé de ces tuples où le numéro de branche est BOO5 (S22), et l'autre composé de ces tuples où le numéro de branche est B007 (S23), comme illustré à la Figure 24.12. Le schéma de fragmentation satisfait les règles de correctionÿ:

- Complétude. Chaque attribut de la relation Staff apparaît dans les fragments S1 ou S2 ; chaque tuple (partiel) apparaît dans le fragment S1 et soit dans le fragment S21, S22 ou S23.

Fragment S_1

staffNo	position	sex	DOB	salary
SL21	Manager	M	1-Oct-45	30000
SG37	Assistant	F	10-Nov-60	12000
SG14	Supervisor	M	24-Mar-58	18000
SA9	Assistant	F	19-Feb-70	9000
SG5	Manager	F	3-Jun-40	24000
SL41	Assistant	F	13-Jun-65	9000

Fragment S_{21}

staffNo	fName	lName	branchNo
SG37	Ann	Beech	B003
SG14	David	Ford	B003
SG5	Susan	Brand	B003

Fragment S_{22}

staffNo	fName	lName	branchNo
SL21	John	White	B005
SL41	Julie	Lee	B005

Fragment S_{23}

staffNo	fName	lName	branchNo
SA9	Mary	Howe	B007

Figure 24.12 Mixed fragmentation of Staff.

- Reconstruction. La relation Staff peut être reconstruite à partir des fragments à l'aide des opérations de jointure Union et Naturalÿ:

  S1 1 (S21 ø S22 ø S23) 5 Mire

- Disjonction. Les fragments sont disjoints ; il ne peut y avoir aucun membre du personnel qui travaille dans plus d'une succursale et S1 et S2 sont disjoints à l'exception de la duplication nécessaire de la clé primaire.

Fragmentation horizontale dérivée Certaines applications peuvent impliquer une jointure de deux relations ou plus. Si les relations sont stockées à des emplacements différents, il peut y avoir une surcharge importante lors du traitement de la jointure. Dans de tels cas, il peut être plus approprié de s'assurer que les relations, ou des fragments de relations, se trouvent au même endroit. Nous pouvons y parvenir en utilisant la fragmentation horizontale dérivée.

Fragment dérivé	Fragment horizontal basé sur la fragmentation horizontale d'une relation parent.

Nous utilisons le terme enfant pour faire référence à la relation qui contient la clé étrangère et parent à la relation contenant la clé primaire ciblée. La fragmentation dérivée est définie en utilisant l'opération Semijoin de l'algèbre relationnelle (voir Section 5.1.3). Étant donné une relation enfant et un parent , la fragmentation dérivée de est définie comme suit : R S R

$$R_i = R \ltimes_f S_i \quad 1 \le i \le w$$

où w est le nombre de fragments horizontaux définis sur et est l'attribut de jointure. S f

EXEMPLE 24.5 Fragmentation horizontale dérivée

Nous pouvons avoir une application qui relie les relations Staff et PropertyForRent. Pour cet exemple, nous supposons que le personnel est fragmenté horizontalement en fonction du numéro d'agence, de sorte que les données relatives à l'agence sont stockées localement :

S3: sbranchNo 5 'B003' (personnel)
S4: sbranchNo 5 'B005' (personnel)
S5: sbranchNo 5 'B007' (personnel)

Nous supposons également que la propriété PG4 est actuellement gérée par SG14. Il serait utile de stocker les données de propriété en utilisant la même stratégie de fragmentation. Ceci est réalisé en utilisant la fragmentation dérivée pour fragmenter horizontalement la relation PropertyForRent en fonction du numéro de brancheÿ:

Pi 5 PropertyForRent 1 staffNo 3 Si i # 5

Cela produit trois fragments (P3, P4 et P5), l'un composé des propriétés gérées par le personnel de l'agence numéro B003 (P3), l'autre composé des propriétés gérées par le personnel de l'agence BOO5 (P4) et l'autre composé de les propriétés gérées par le personnel de la succursale B007 (P5), comme le montre la figure 24.13. On peut facilement montrer que ce schéma de fragmentation satisfait les règles de correction. Nous laissons cela comme un exercice pour le lecteur.

Si une relation contient plus d'une clé étrangère, il sera nécessaire de sélectionner l'une des relations référencées comme parent. Le choix peut être basé sur la fragmentation utilisée le plus fréquemment ou la fragmentation avec de meilleures caractéristiques de jointure, c'est-à-dire la jointure impliquant des fragments plus petits ou la jointure qui peut être effectuée en parallèle à un degré plus élevé.

Fragment P_3

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	SG14
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14

Fragment P_4

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41

Fragment P_5

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9

Illustration 24.13 Derived fragmentation of PropertyForRent based on Staff.

Pas de fragmentation

Une dernière stratégie consiste à ne pas fragmenter une relation. Par exemple, la relation Branch ne contient qu'un petit nombre de tuples et n'est pas mise à jour très fréquemment. Plutôt que d'essayer de fragmenter horizontalement la relation sur le numéro de branche par exemple, il serait plus judicieux de laisser la relation entière et de simplement répliquer la branche relation sur chaque site.

Résumé d'une méthodologie de conception de bases de données distribuées

Nous sommes maintenant en mesure de résumer une méthodologie de conception de bases de données distribuées.

(1) Utilisez la méthodologie décrite dans les chapitres 16 et 17 pour produire une conception des relations globales.

(2) De plus, examinez la topologie du système. Par exemple, considérez si DreamHome aura une base de données dans chaque succursale, ou dans chaque ville, ou éventuellement au niveau régional. Dans le premier cas, la fragmentation des relations sur la base du numéro de branche peut être appropriée. Cependant, dans ces deux derniers cas, il peut être plus approprié d'essayer de fragmenter les relations sur une base de ville ou de région.

(3) Analysez les transactions les plus importantes du système et identifiez où une fragmentation zonale ou verticale peut être appropriée.

(4) Décidez quelles relations ne doivent pas être fragmentées — ces relations se reproduiront partout. Dans le diagramme ER global, supprimez les relations qui ne seront pas fragmentées et toutes les relations dans lesquelles ces transactions sont impliquées.

(5) Examiner les relations qui sont d'un côté d'une relation et décider d'un schéma de fragmentation approprié pour ces relations, en tenant compte

la topologie du système. Les relations sur le côté multiple d'une relation peuvent être candidats à la fragmentation dérivée.

(6) Au cours de l'étape précédente, vérifiez les situations où soit vertical ou mixte la fragmentation serait appropriée (c'est-à-dire lorsque les transactions nécessitent un accès à un sous-ensemble des attributs d'une relation).

24.5 Transparents dans un DDBMS

La définition d'un DDBMS donnée à la section 24.1.1 indique que le système doit rendre la distribution transparente pour l'utilisateur. La transparence cache la mise en œuvre détails de l'utilisateur. Par exemple, dans un SGBD centralisé, l'indépendance des données est une forme de transparence — elle masque les changements dans la définition et l'organisation du données de l'utilisateur. Un DDBMS peut fournir différents niveaux de transparence. Cependant, ils participent tous au même objectif global : faire usage de la distribution base de données équivalente à celle d'une base de données centralisée. Nous pouvons identifier quatre principaux types de transparence dans un DDBMSÿ:

- transparence de la distributionÿ;
- transparence des transactionsÿ;
- transparence des performancesÿ;
- Transparence du SGBD.

Avant de discuter de chacune de ces transparences, il convient de noter que la transparence n'est pas un objectif universellement accepté. Par exemple, Gray (1989) soutient que la transparence totale rend la gestion des données distribuées très difficile et que les applications codées avec un accès transparent à des bases de données géographiquement distribuées ont une gérabilité médiocre, une faible modularité et un message médiocre performance. Notez que toutes les transparences dont nous discutons sont rarement satisfaites par un seul système.

24.5.1 Transparence de distribution

La transparence de la distribution permet à l'utilisateur de percevoir la base de données comme une seule entité logique. Si un DDBMS présente une distribution transparente, l'utilisateur ne besoin de savoir que les données sont fragmentées (transparence de la fragmentation) ou l'emplacement des éléments de données (transparence de localisation).

Si l'utilisateur a besoin de savoir que les données sont fragmentées et l'emplacement des fragments, nous appelons cette transparence de cartographie locale. Ces transparences sont commandé, comme nous en discutons maintenant. Pour illustrer ces concepts, considérons la distribution de la relation Staff donnée dans l'exemple 24.4, telle que :

S1ÿ: PstaffNo, poste, sexe, date de naissance, salaire (personnel)	situé sur le site 5
S2 : PstaffNo, fName, lName, branchNo(Personnel)	
S21ÿ: branche n° 5 'B003' (S2)	situé sur le site 3
S22ÿ: branche n° 5 'B005' (S2)	situé sur le site 5
S23ÿ: branche n° 5 'B007' (S2)	situé au site 7

Transparence de la fragmentation La

fragmentation est le plus haut niveau de transparence de la distribution. Si la transparence de la fragmentation est fournie par le DDBMS, l'utilisateur n'a pas besoin de savoir que les données sont fragmentées. Par conséquent, les accès à la base de données sont basés sur le schéma global, de sorte que l'utilisateur n'a pas besoin de spécifier les noms des fragments ou les emplacements des données. Par exemple, pour récupérer les noms de tous les Managers, avec une transparence de fragmentation on pourrait écrire :

```
SELECT fName, lName
DE Personnel
WHERE position 5 'Gestionnaire' ;
```

Il s'agit de la même instruction SQL que nous écririons dans un système centralisé.

Transparence de l'emplacement

L'emplacement est le niveau intermédiaire de transparence de la distribution. Avec la transparence de l'emplacement, l'utilisateur doit savoir comment les données ont été fragmentées mais n'a toujours pas besoin de connaître l'emplacement des données. La requête précédente devient maintenant sous la transparence de l'emplacementÿ:

```
SELECT fName, lName
À PARTIR DE S21
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position 5 'Manager')
    SYNDICAT
SELECT fName, lName
A PARTIR DE S22
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position 5 'Manager')
    SYNDICAT
SELECT fName, lName
À PARTIR DE S23
WHERE staffNo IN (SELECT staffNo FROM S1 WHERE position 5 'Manager');
```

Nous devons maintenant spécifier les noms des fragments dans la requête. Nous devons également utiliser une jointure (ou une sous-requête), car les attributs position et fName/lName apparaissent dans des fragments verticaux différents. Le principal avantage de la transparence de l'emplacement est que la base de données peut être physiquement réorganisée sans impact sur les programmes d'application qui y accèdent.

Transparence de la réplication La

transparence de la réplication est étroitement liée à la transparence de l'emplacement, ce qui signifie que l'utilisateur n'est pas conscient de la réplication des fragments. La transparence de la réplication est impliquée par la transparence de l'emplacement. Cependant, il est possible qu'un système n'ait pas de transparence de localisation mais ait une transparence de réplication.

Transparence du mappage local Il s'agit

du niveau le plus bas de transparence de la distribution. Avec la transparence du mappage local, l'utilisateur doit spécifier à la fois les noms des fragments et l'emplacement des éléments de données,

en tenant compte de toute réplication qui pourrait exister. Sous transparence de mappage local, l'exemple de requête devientÿ:

```
SELECT fName, lName
DEPUIS S21 AU SITE 3
WHERE staffNo IN (SELECT staffNo FROM S1 AT SITE 5 WHERE position 5 'Manager')
    UNION
SELECT fName, lName
DEPUIS S22 AU SITE 5
WHERE staffNo IN (SELECT staffNo FROM S1 AT SITE 5 WHERE position 5 'Manager')
    UNION
SELECT fName, lName
DEPUIS S23 AU SITE 7
WHERE staffNo IN (SELECT staffNo FROM S1 AT SITE 5 WHERE position 5'Manager');
```

À des fins d'illustration, nous avons étendu SQL avec le mot-clé AT SITE pour exprimer où se trouve un fragment particulier. De toute évidence, il s'agit d'une requête plus complexe et plus longue à saisir pour l'utilisateur que les deux premières. Il est peu probable qu'un système qui n'offrait que ce niveau de transparence soit acceptable pour
les utilisateurs finaux.

Transparence de nommage

Corollaire des transparences de distribution précédentes, nous avons la transparence de nommage. Comme dans une base de données centralisée, chaque élément d'une base de données distribuée doit avoir un nom unique. Par conséquent, le DDBMS doit s'assurer que deux sites ne créent pas un objet de base de données portant le même nom. Une solution à ce problème consiste à créer un serveur de noms central, qui a la responsabilité d'assurer l'unicité de tous les noms dans le système. Cependant, cette approche se traduit par :

- perte d'une certaine autonomie locale ;
- des problèmes de performances, si le site central devient un goulot d'étranglement ;
- Faible Disponibilité; si le site central tombe en panne, les sites restants ne peuvent pas créer de nouveaux objets de base de données.

Une solution alternative consiste à préfixer un objet avec l'identifiant du site qui l' a créé. Par exemple, la relation Branche créée sur le site S1 peut être nommée S1.Branch. De même, il faut pouvoir identifier chaque fragment et chacune de ses copies. Ainsi, la copie 2 du fragment 3 de la relation Branch créée sur le site S1 peut être appelée S1.Branch.F3.C2. Cependant, cela entraîne une perte de transparence de la distribution.

Une approche qui résout les problèmes avec ces deux solutions utilise des alias (parfois appelés synonymes) pour chaque objet de base de données. Ainsi, S1.Branch.F3.C2 peut être appelé LocalBranch par l'utilisateur du site S1. Le DDBMS a pour tâche de mapper un alias à l'objet de base de données approprié.

Le système distribué R* fait la distinction entre le nom d'impression d'un objet et son nom à l'échelle du système. Le printname est le nom que les utilisateurs utilisent normalement pour faire référence à l'objet. Le nom à l'échelle du système est un identifiant interne unique au monde pour l'objet qui est garanti de ne jamais changer. Le nom à l'échelle du système est composé de quatre composantsÿ:

- Identifiant du créateur. Un identifiant de site unique pour l'utilisateur qui a créé l'objetÿ;
- ID du site du créateur. Un identifiant global unique pour le site à partir duquel l'objet a été créé ÿ; Nom local. Un nom non qualifié pour l'objetÿ; ID du lieu de naissance. Un identifiant global unique pour le
- site sur lequel l'objet a été initialement stocké (comme nous l'avons vu pour le catalogue global du
- système dans la section 24.3.4).

Par exemple, le nom à l'échelle du systèmeÿ:

Manager@London.LocalBranch@Glasgow

représente un objet avec le nom local LocalBranch, créé par l'utilisateur Manager sur le site de Londres et initialement stocké sur le site de Glasgow.

24.5.2 Transparence des transactions La transparence des

transactions dans un environnement DDBMS garantit que toutes les transactions distribuées maintiennent l' intégrité et la cohérence de la base de données distribuée. Une transaction distribuée accède à des données stockées à plusieurs endroits. Chaque transaction est divisée en un certain nombre de sous-transactions, une pour chaque site auquel il faut accéderÿ; une sous-transaction est représentée par un agent, comme illustré dans l'exemple suivant.

EXEMPLE 24.6 Transaction distribuée

Considérez une transaction T qui imprime les noms de tout le personnel, en utilisant le schéma de fragmentation défini précédemment comme S1, S2, S21, S22 et S23. Nous pouvons définir trois sous-transactions TS3 , TS5 et TS7 pour représenter les agents aux sites 3, 5 et 7, respectivement. Chaque sous-transaction imprime les noms du personnel de ce site. La transaction distribuée est illustrée à la Figure 24.14. Notez le parallélisme inhérent au systèmeÿ: les sous-transactions de chaque site peuvent s'exécuter simultanément.

Time	T_{s_3}	T_{s_5}	T_{s_7}
t_1	begin_transaction	begin_transaction	begin_transaction
t_2	read(fName, lName)	read(fName, lName)	read(fName, lName)
t_3	print(fName, lName)	print(fName, lName)	print(fName, lName)
t_4	end_transaction	end_transaction	end_transaction

Figure 24.14 Distributed transaction.

L'atomicité de la transaction distribuée est toujours fondamentale pour le concept de transaction, mais en plus le DDBMS doit aussi assurer l'atomicité de chaque sous-transaction (voir section 22.1.1). Par conséquent, non seulement le DDBMS doit assurer la synchronisation des sous-transactions avec d'autres transactions locales qui s'exécutent simultanément sur un site, mais il doit également assurer la synchronisation des sous-transactions avec des transactions globales s'exécutant simultanément sur le même site ou sur des sites différents. La transparence des transactions dans un SGBD distribué est compliquée par les schémas de fragmentation, d'allocation et de réplication. Nous considérons deux autres aspects de la transparence des transactions : la transparence de la concurrence et la transparence des défaillances.

Transparence de la concurrence

La transparence de la concurrence est fournie par le DDBMS si les résultats de toutes les transactions simultanées (distribuées et non distribuées) s'exécutent indépendamment et sont logiquement cohérents avec les résultats obtenus si les transactions sont exécutées une à la fois, dans un ordre de série arbitraire. Ce sont les mêmes principes fondamentaux que ceux que nous avons évoqués pour le SGBD centralisé à la section 22.2.2. Cependant, il y a la complexité supplémentaire que le DDBMS doit garantir que les transactions globales et locales n'interfèrent pas les unes avec les autres. De même, le DDBMS doit assurer la cohérence de toutes les sous-transactions de la transaction globale.

La réplication rend la question de la concurrence plus complexe. Si une copie d'un élément de données répliqué est mise à jour, la mise à jour doit finalement être propagée à toutes les copies. Une stratégie évidente consiste à propager les modifications dans le cadre de la transaction d'origine, ce qui en fait une opération atomique. Cependant, si l'un des sites détenant une copie n'est pas joignable au moment du traitement de la mise à jour, parce que le site ou le lien de communication a échoué, la transaction est retardée jusqu'à ce que le site soit joignable. S'il existe de nombreuses copies de l'élément de données, la probabilité de réussite de la transaction diminue de manière exponentielle. Une autre stratégie consiste à limiter la propagation des mises à jour aux sites actuellement disponibles. Les sites restants doivent être mis à jour lorsqu'ils redeviennent disponibles. Une autre stratégie consisterait à permettre aux mises à jour des copies de se produire de manière asynchrone, quelque temps après la mise à jour d'origine. Le délai de retour à la consistance peut aller de quelques secondes à plusieurs heures. Nous expliquons comment gérer correctement le contrôle de concurrence distribuée et la réplication dans le chapitre suivant.

Transparence des

défaillances Dans la section 22.3.2, nous avons indiqué qu'un SGBD centralisé doit fournir un mécanisme de récupération qui garantit qu'en présence de défaillances, les transactions sont atomiques : soit toutes les opérations de la transaction sont effectuées, soit aucune. De plus, une fois qu'une transaction a été validée, les modifications sont durables. Nous avons également examiné les types de pannes pouvant survenir dans un système centralisé, telles que les pannes système, les pannes de support, les erreurs logicielles, la négligence, les catastrophes physiques naturelles et le sabotage. Dans l'environnement distribué, le DDBMS doit également prendre en chargeÿ:

- la perte d'un message; la
- défaillance d'un lien de communicationÿ; la
- défaillance d'un site ; partitionnement du
- réseau.

Le DDBMS doit assurer l'atomicité de la transaction globale, ce qui signifie s'assurer que les sous-transactions de la transaction globale sont toutes validées ou toutes abandonnées. Ainsi, le DDBMS doit synchroniser la transaction globale pour s'assurer que toutes les sous-transactions se sont terminées avec succès avant d'enregistrer un COMMIT final pour la transaction globale. Par exemple, considérons une transaction globale qui doit mettre à jour des données sur deux sites, disons S1 et S2. La sous-transaction sur le site S1 se termine avec succès et est validée, mais la sous-transaction sur le site S2 est incapable de valider et annule les modifications pour assurer la cohérence locale. La base de données distribuée est maintenant dans un état incohérentÿ: nous ne pouvons pas annuler la validation des données sur le site S1, en raison de la

propriété de durabilité de la sous-transaction à S1. Nous expliquons comment gérer correctement la récupération de base de données distribuée dans le chapitre suivant.

Classement des opérations

Avant de terminer notre discussion sur les transactions dans ce chapitre, nous présentons brièvement une classification des transactions définies dans l'architecture de base de données relationnelle distribuée (DRDA) d'IBM. Dans DRDA, il existe quatre types de transactions, chacune avec un niveau de complexité progressif dans l'interaction entre les SGBDÿ:

(1) demande à distanceÿ;

(2) unité de travail éloignée; (3)

unité de travail distribuée; (4)

demande distribuée.

Dans ce contexte, une « requête » équivaut à une instruction SQL et une « unité de travail » est une transaction. Les quatre niveaux sont illustrés à la Figure 24.15.

(1) Requête à distance. Une application sur un site peut envoyer une requête (instruction SQL) à un site distant pour exécution. La demande est entièrement exécutée sur le site distant et ne peut référencer des données que sur le site distant.

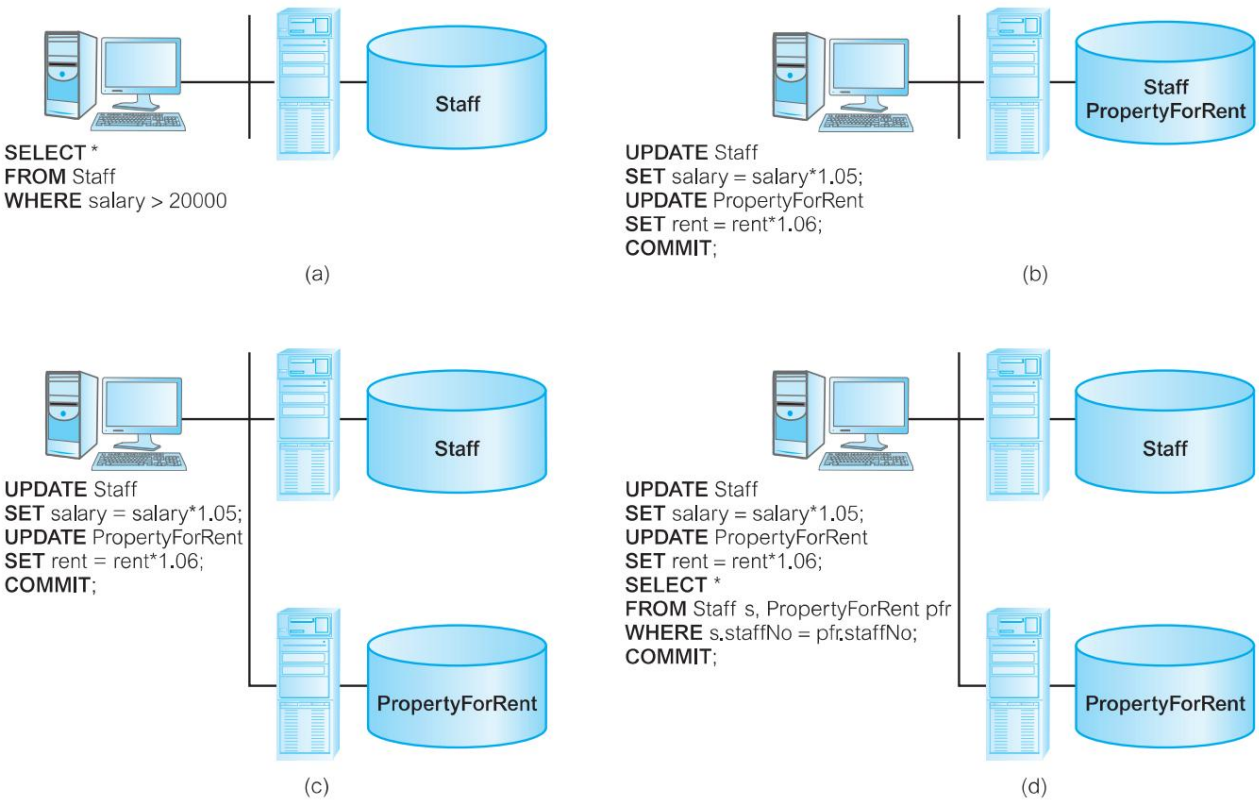

Image 24.15 DRDA classification of transactions: (a) remote request; (b) remote unit of work; (c) distributed unit of work; (d) distributed request.

(2) Unité de travail à distance. Une application sur un site (local) peut envoyer toutes les instructions SQL d'une unité de travail (transaction) à un site distant pour exécution. Toutes les instructions SQL sont entièrement exécutées sur le site distant et ne peuvent référencer des données que sur le site distant. Cependant, le site local décide si la transaction doit être validée ou annulée.

(3) Unité de travail distribuée. Une application sur un site (local) peut envoyer une partie ou la totalité des instructions SQL d'une transaction à un ou plusieurs sites distants pour exécution. Chaque instruction SQL est entièrement exécutée sur le site distant et ne peut référencer des données que sur le site distant. Cependant, différentes instructions SQL peuvent être exécutées sur différents sites. Là encore, le site local décide si la transaction doit être validée ou annulée.

(4) Requête distribuée. Une application sur un site (local) peut envoyer une partie ou la totalité des instructions SQL d'une transaction à un ou plusieurs sites distants pour exécution. Cependant, une instruction SQL peut nécessiter l'accès aux données de plusieurs sites (par exemple, l'instruction SQL peut avoir besoin de joindre ou d'union des relations/fragments situés sur différents sites).

24.5.3 Transparence des performances

La transparence des performances nécessite qu'un DDBMS fonctionne comme s'il s'agissait d'un SGBD centralisé. Dans un environnement distribué, le système ne doit subir aucune dégradation des performances due à l'architecture distribuée, telle que la présence du réseau. La transparence des performances nécessite également que le DDBMS détermine la stratégie la plus rentable pour exécuter une demande.

Dans un SGBD centralisé, le processeur de requêtes (QP) doit évaluer chaque demande de données et trouver une stratégie d'exécution optimale, consistant en une séquence ordonnée d'opérations sur la base de données. Dans un environnement distribué, le processeur de requêtes distribué (DQP) mappe une demande de données dans une séquence ordonnée d'opérations sur les bases de données locales. Il a la complexité supplémentaire de prendre en compte les schémas de fragmentation, de réplication et d'allocation. Le DQP doit décider :

- à quel fragment accéderÿ;
- quelle copie d'un fragment utiliser, si le fragment est répliquéÿ; quel
- emplacement utiliser.

Le DQP produit une stratégie d'exécution optimisée par rapport à une fonction de coût. En règle générale, les coûts associés à une demande distribuée incluentÿ:

- le coût du temps d'accès (E/S) impliqué dans l'accès aux données physiques sur
- disqueÿ; le coût en temps CPU encouru lors de l'exécution d'opérations sur des données dans
- la mémoire principaleÿ; le coût de communication associé à la transmission de données sur le réseau.

Les deux premiers facteurs sont les seuls pris en compte dans un système centralisé. Dans un environnement distribué, le DDBMS doit tenir compte du coût de communication, qui peut être le facteur le plus important dans les WAN avec une bande passante de quelques kilo-octets par seconde. Dans de tels cas, l'optimisation peut ignorer les coûts d'E/S et de CPU. Cependant, les réseaux locaux ont une bande passante comparable à celle des disques, donc dans de tels cas, l'optimisation ne doit pas ignorer entièrement les coûts d'E/S et de CPU.

Une approche de l'optimisation des requêtes minimise le coût total du temps qui sera engagé dans l'exécution de la requête (Sacco et Yao, 1982). Une approche alternative

minimise le temps de réponse de la requête, auquel cas le DQP tente de maximiser l'exécution parallèle des opérations (Epstein et al., 1978). Parfois, le temps de réponse sera nettement inférieur au coût total du temps. L'exemple suivant, adapté de Rothnie et Goodman (1977), illustre la grande variation des temps de réponse qui peut résulter de stratégies d'exécution différentes mais plausibles.

EXEMPLE 24.7 Traitement distribué des requêtes

Considérons un schéma relationnel DreamHome simplifié composé des trois relations suivantesÿ:

Property(propertyNo, city) 10 000 enregistrements stockés à Londres 100 000
Client(n°client, prixmax) enregistrements stockés à Glasgow
Affichage(propertyNo, clientNo) 1 000 000 enregistrements stockés à Londres

Pour répertorier les propriétés à Aberdeen qui ont été consultées par des clients dont la limite de prix maximale est supérieure à 200ÿ000ÿ£, nous pouvons utiliser la requête SQL suivanteÿ:

```
SELECT p.propertyNo
FROM Propriété p INNER JOIN
        (Client c INNER JOIN Affichage v ON c.clientNo 5 v.clientNo)
ON p.propertyNo 5 v.propertyNo
WHERE p.city 5 'Aberdeen' AND c.maxPrice . 200000ÿ;
```

Pour simplifier, supposons que chaque tuple dans chaque relation ait une longueur de 100 caractères, qu'il y ait 10 clients avec un prix maximum supérieur à 200 000 £, qu'il y ait 100 000 visites de propriétés à Aberdeen et que le temps de calcul soit négligeable par rapport au temps de communication. Nous supposons en outre que le système de communication a un débit de transmission de données de 10 000 caractères par seconde et un délai d'accès de 1 seconde pour envoyer un message d'un site à un autre.

Rothnie identifie six stratégies possibles pour cette requête, comme résumé dans le tableau 24.4. En utilisant l'algorithme pour le temps de communication donné dans la section 24.2, nous calculons les temps de réponse pour ces stratégies comme suit :

TABLEAU 24.4 Comparison of distributed query processing strategies.

STRATÉGIE	TEMPS
(1) Move Client relation to London and process query there	16.7 minutes
(2) Move Property and Viewing relations to Glasgow and process query there	28 hours
(3) Join Property and Viewing relations at London, select tuples for Aberdeen properties and, for each of these in turn, check at Glasgow to determine if associated maxPrice ■£200,000	2.3 days
(4) Select clients with maxPrice ■£200,000 at Glasgow and, for each one found, check at London for a viewing involving that client and an Aberdeen property	20 seconds
(5) Join Property and Viewing relations at London, select Aberdeen properties, project result over propertyNo and clientNo, and move this result to Glasgow for matching with maxPrice ■£200,000	16.7 minutes
(6) Select clients with maxPrice ■£200,000 at Glasgow and move the result to London for matching with Aberdeen properties	1 second

Stratégieÿ1ÿ: Déplacer la relation client vers Londres et y traiter la requêteÿ:

Temps 5 1 1 (100 000 * 100/10 000) ù 16,7 minutes

Stratégie 2ÿ: Déplacez les relations de propriété et de visualisation vers Glasgow et traitez-y la requêteÿ:

Temps 5 2 1 [(1 000 000 1 10 000) * 100/10 000] ù 28 heures

Stratégie 3 : Rejoindre les relations Property et Viewing à Londres, sélectionner des tuples pour les propriétés d'Aberdeen, puis pour chacun de ces tuples, vérifier à Glasgow pour déterminer si le maxPrice du client associé > 200 000 £. La vérification de chaque tuple implique deux messagesÿ: une requête et une réponse.

Temps 5 100 000 * (1 1 100/10 000) 1 100 000 * 1 ù 2,3 jours

Stratégie 4 : sélectionnez les clients avec maxPrice . 200 000 £ à Glasgow et pour chacun trouvé, vérifiez à Londres s'il y a une visite impliquant ce client et une propriété d'Aberdeen. Encore une fois, deux messages sont nécessairesÿ:

Temps 5 10 * (1 1 100/10 000) 1 10 * 1 ù 20 secondes

Stratégie 5ÿ: Joindre les relations Property et Viewing à Londres, sélectionner les propriétés d'Aberdeen, le résultat du projet sur propertyNo et clientNo, et déplacer ce résultat vers Glasgow pour qu'il corresponde à maxPrice . 200 000 £. Pour plus de simplicité, nous supposons que le résultat projeté comporte toujours 100ÿcaractèresÿ:

Temps 5 1 1 (100 000 * 100/10 000) ù 16,7 minutes

Stratégie 6 : Sélectionnez les clients avec maxPrice . 200 000 £ à Glasgow et déplacer le résultat à Londres pour une correspondance avec les propriétés d'Aberdeenÿ:

Temps 5 1 1 (10 * 100/10 000) ù 1 seconde

Les temps de réponse varient de 1 seconde à 2,3 jours, mais chaque stratégie est un moyen légitime d'exécuter la requête. De toute évidence, si la mauvaise stratégie est choisie, l'effet peut être dévastateur sur les performances du système. Nous discuterons plus en détail du traitement distribué des requêtes dans la section 25.6.

24.5.4 Transparence du SGBD

La transparence du SGBD cache la connaissance que les SGBD locaux peuvent être différents et n'est donc applicable qu'aux SDDB hétérogènes. C'est l'une des transparences les plus difficiles à fournir comme généralisation. Nous avons discuté des problèmes associés à la fourniture de systèmes hétérogènes à la section 24.1.3.

24.5.5 Résumé des transparences dans un DDBMS

Au début de cette section sur les transparences dans un DDBMS, nous avons mentionné que la transparence totale n'est pas un objectif universellement accepté. Comme vous l'avez vu, la transparence n'est pas un concept de « tout ou rien », mais elle peut être assurée à différents niveaux. Chaque niveau nécessite un type particulier d'accord entre les sites participants. Par exemple, en toute transparence, les sites doivent s'accorder sur des éléments tels que le modèle de données, l'interprétation des schémas, la représentation des données et les fonctionnalités fournies par chaque site. À l'autre extrémité du spectre, dans un système non transparent, il n'y a accord que sur le format d'échange de données et les fonctionnalités fournies par chaque site.

Du point de vue de l'utilisateur, une transparence complète est hautement souhaitable. Cependant, du point de vue de l'administrateur de base de données local, un accès totalement transparent peut être difficile à contrôler. En tant que mécanisme de sécurité, la fonction de visualisation traditionnelle peut ne pas être assez puissante pour fournir une protection suffisante. Par exemple, le mécanisme de vue SQL permet de restreindre l'accès à une relation de base, ou à un sous-ensemble d'une relation de base, à des utilisateurs nommés, mais il ne permet pas facilement de restreindre l'accès en fonction d'un ensemble de critères autres que le nom d'utilisateur. Dans l'étude de cas DreamHome, nous pouvons restreindre l'accès à la suppression de la relation de location aux membres du personnel nommés, mais nous ne pouvons pas facilement empêcher la suppression d'un contrat de location si le bail est terminé, que tous les paiements en souffrance ont été effectués par le locataire et que le la propriété est encore dans un état satisfaisant.

Nous pouvons trouver plus facile de fournir ce type de fonctionnalité dans une procédure appelée à distance. De cette façon, les utilisateurs locaux peuvent voir les données qu'ils sont normalement autorisés à voir à l'aide des mécanismes de sécurité standard du SGBD. Cependant, les utilisateurs distants ne voient que les données encapsulées dans un ensemble de procédures, de la même manière que dans un système orienté objet. Ce type d'architecture fédérée est plus simple à mettre en œuvre qu'une transparence complète et peut fournir un plus grand degré d'autonomie locale.

24.6 Les Douze Règles de Date pour un DDBMS

Dans cette dernière section, nous listons les douze règles (ou objectifs) de Date pour les DDBMS (Date, 1987b). La base de ces règles est qu'un SGBD distribué devrait ressembler à un SGBD non distribué pour l'utilisateur. Ces règles s'apparentent aux douze règles de Codd pour les systèmes relationnels, présentées à l'annexe G.

Principe fondamental Pour

l'utilisateur, un système distribué doit ressembler exactement à un système non distribué.

(1) Autonomie locale Les

sites d'un système distribué doivent être autonomes. Dans ce contexte, l'autonomie signifie que :

- les données locales sont détenues et gérées
- localementÿ; les opérations locales restent purement
- locales ; toutes les opérations sur un site donné sont contrôlées par ce site.

(2) Ne pas dépendre d'un site central Il ne

devrait y avoir aucun site sans lequel le système ne peut pas fonctionner. Cela implique qu'il ne devrait pas y avoir de serveurs centraux pour des services tels que la gestion des transactions, la détection des interblocages, l'optimisation des requêtes et la gestion du catalogue système global.

(3) Fonctionnement continu

Idéalement, un arrêt planifié du système ne devrait jamais être nécessaire pour des opérations telles queÿ:

- ajouter ou supprimer un site du systèmeÿ; la création et
- la suppression dynamiques de fragments sur un ou plusieurs sites.

(4) Indépendance de l' emplacement

L'indépendance de l'emplacement équivaut à la transparence de l'emplacement. L'utilisateur doit pouvoir accéder à la base de données à partir de n'importe quel site. En outre, l'utilisateur doit pouvoir accéder à toutes les données comme si elles étaient stockées sur le site de l'utilisateur, quel que soit l'endroit où elles sont physiquement stockées.

(5) Indépendance à la fragmentation L'utilisateur

doit pouvoir accéder aux données, quelle que soit leur fragmentation.

(6) Indépendance de la réplication

L'utilisateur ne doit pas savoir que les données ont été répliquées. Ainsi, l'utilisateur ne devrait pas être en mesure d'accéder directement à une copie particulière d'un élément de données et l'utilisateur ne devrait pas non plus avoir à mettre à jour spécifiquement toutes les copies d'un élément de données.

(7) Traitement distribué des requêtes Le système

doit être capable de traiter les requêtes qui font référence à des données sur plus d'un site.

(8) Traitement distribué des transactions Le système

doit prendre en charge la transaction en tant qu'unité de récupération. Le système doit garantir que les transactions globales et locales sont conformes aux règles ACID pour les transactionsÿ: atomicité, cohérence, isolation et durabilité.

(9) Indépendance matérielle Il devrait être

possible d'exécuter le DDBMS sur une variété de plates-formes matérielles.

(10) Indépendance du système d'exploitation Comme

corollaire de la règle précédente, il devrait être possible d'exécuter le DDBMS sur une variété de systèmes d'exploitation.

(11) Indépendance du réseau Encore une

fois, il devrait être possible d'exécuter le DDBMS sur une variété de réseaux de communication disparates.

(12) Indépendance de la base de données

Il devrait être possible d'avoir un DDBMS composé de différents SGBD locaux, prenant peut-être en charge différents modèles de données sous-jacents. En d'autres termes, le système doit supporter l'hétérogénéité.

Les quatre dernières règles sont des idéaux. Parce que les règles sont si générales et qu'il y a un manque de normes dans les architectures informatiques et réseau, nous ne pouvons nous attendre qu'à une conformité partielle des fournisseurs dans un avenir prévisible.

Résumé du chapitre

- A base de données distribuée is a logically interrelated collection of shared data (and a description of this data), physically distributed over a computer network. The DDBMS is the software that transparently manages the distributed database.
- A DDBMS is distinct from SGBD parallèle à , where a centralized DBMS is accessed over a network. It is also distinct from a traitement distribuéwhich is a DBMS running across multiple processors and disks and which has been designed to evaluate operations in parallel, whenever possible, in order to improve performance.
- The advantages of a DDBMS are that it reflects the organizational structure; it makes remote data more shareable; it improves reliability, availability, and performance; it may be more economical; it provides for modular growth, facilitates integration, and helps organizations remain competitive. The major disadvantages are cost, complexity, lack of standards, and experience.
- A DDBMS may be classified as homogeneous or heterogeneous. In a homogène system, all sites use the same DBMS product. In a hétérogène system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical, and object-oriented DBMSs.
- A système multibase de données (MDBS) is a distributed DBMS in which each site maintains complete autonomy. An MDBS resides transparently on top of existing database and file systems and presents a single database to its users. It maintains a global schema against which users issue queries and updates; an MDBS maintains only the global schema and the local DBMSs themselves maintain all user data.
- Communication takes place over a network, which may be a local area network (LAN) or a wide area network (WAN). LANs are intended for short distances and provide faster communication than WANs. A special case of the WAN is a metropolitan area network (MAN), which generally covers a city or suburb.
- As well as having the standard functionality expected of a centralized DBMS, a DDBMS will need extended communication services, extended system catalog, distributed query processing, and extended security, con currency, and recovery services.
- A relation may be divided into a number of subrelations called fragments , which are attribué to one or more sites. Fragments may be répliqué to provide improved availability and performance.
- There are two main types of fragmentation: horizontal and vertical . Horizontal fragments are subsets of tuples and vertical fragments are subsets of attributes. There are also two other types of fragmentation: mixte and dérivé , a type of horizontal fragmentation where the fragmentation of one relation is based on the fragmentation of another relation.
- The definition and allocation of fragments are carried out strategically to achieve locality of reference, improved reliability and availability, acceptable performance, balanced storage capacities and costs, and minimal communication costs. The three correctness rules of fragmentation are completeness, reconstruction, and disjointness.
- There are four allocation strategies regarding the placement of data: réplication (a single centralized database), fragmenté (fragments assigned to one site), complète centralisée réplication(complete copy of the database maintained at each site), and sélective (combination of the first three).
- The DDBMS should appear like a centralized DBMS by providing a series of transparencies. With transparence de la distribution , users should not know that the data has been fragmented/replicated. With transparence des transactions, the consistency of the global database should be maintained when multiple users are access ing the database concurrently and when failures occur. With transparence des performances, the system should be able to efficiently handle queries that reference data at more than one site. With SGBD transparence , it should be possible to have different DBMSs in the system.

Questions de révision

24.1 Explain what is meant by a DDBMS and discuss the motivation in providing such a system.

24.2 Compare and contrast a DDBMS with distributed processing. Under what circumstances would you choose a DDBMS over distributed processing?

24.3 Discuss why processes used to design a centralized relational database are not the same as those for a distributed relational database.

24.4 Discuss the advantages and disadvantages of a DDBMS.

24,5 What is the difference between a homogeneous DDBMS and a heterogeneous DDBMS? Under what circumstances would such systems generally arise?

24.6 What are network protocols and why are they important?

24,7 What functionality do you expect in a DDBMS?

24,8 What is a multidatabase system? Describe a reference architecture for such a system.

24,9 One problem area with DDBMSs is that of distributed database design. Discuss the issues that have to be addressed with distributed database design. Discuss how these issues apply to the global system catalog.

24.10 What are the differences between horizontal and vertical fragmentation schemes?

24.11 Describe alternative schemes for fragmenting a global relation. State how you would check for correctness to ensure that the database does not undergo semantic change during fragmentation.

24.12 What are the differences between query optimization processes under centralized and distributed DBMSs. Why is the latter thought to be complex?

24.13 A DDBMS must ensure that no two sites create a database object with the same name. One solution to this problem is to create a central name server. What are the disadvantages with this approach? Propose an alternative approach that overcomes these disadvantages.

24.14 What are parallel database management systems? Describe their architecture and the way they are applied in organizations.

Des exercices

A multinational engineering company has decided to distribute its project management information at the regional level in mainland Britain. The current centralized relational schema is as follows:

Employé (NIN, fName, lName, adresse, DOB, sexe, salaire, taxCode, deptNo)
Service (deptNo, deptName, managerNIN, businessAreaNo, regionNo)
Projet (projNo, projName, contractPrice, projectManagerNIN, deptNo)
Travaille sur (NIN, projNo, heures travaillées)
Entreprise (businessAreaNo, businessAreaName)
Région (regionNo, regionName)

where
- **Employé** contains employee details and the national insurance number NIN is the key.
- **département** contains department details and deptNo is the key. gérantNIN identifies the employee who is the manager of the department. There is only one manager for each department.
- **Projet** contains details of the projects in the company and the key is projNo. The project manager is identified by the chef de projetNIN, and the department responsible for the project by deptNo.

Travaille sur contains details of the hours worked by employees on each project and (NIN, projetNo) forms the key.
Affaires contains names of the business areas and the key is businessAreaNo .
Région contains names of the regions and the key is regionNo .

Departments are grouped regionally as follows:

Region 1: Scotland Region 2: Wales Region 3: England

Information is required by business area, which covers: Software Engineering, Mechanical Engineering, and Electrical Engineering. There is no Software Engineering in Wales and all Electrical Engineering departments are in England. Projects are staffed by local department offices.

As well as distributing the data regionally, there is an additional requirement to access the employee data either by personal information (by HR) or by work related information (by Payroll).

24h15 Draw an Entity–Relationship (ER) diagram to represent this system.

24.16 Using the ER diagram from Exercise 24.15, produce a distributed database design for this system, and include:
(un) a suitable fragmentation schema for the system;
(b) in the case of primary horizontal fragmentation, a minimal set of predicates;
(c) the reconstruction of global relations from fragments.

State any assumptions necessary to support your design.

24.17 Repeat Exercise 24.16 for the Maison de rêve case study documented in Appendix A.

24.18 Repeat Exercise 24.16 for the École d'automobile EasyDrive case study documented in Appendix B.2.

24.19 Prepare an action plan for the process of distributing the Wellmeadows case study database documented in Appendix B.3.

24h20 In Section 24.5.1, when discussing naming transparency, we proposed the use of aliases to uniquely identify each replica of each fragment. Provide an outline design for the implementation of this approach to naming transparency.

24.21 Compare a distributed DBMS to which you have access against Date's twelve rules for a DDBMS. For each rule with which the system is not compliant, give reasons why you think there is no conformance to this rule.