Chapitre 4

Architectures d'Espaces de Travail

I. Introduction

Le chapitre précédent formalise l'Espace de Travail et les services qu'il propose. Ces services exportent l'image de la base de données vers le client ou effectuent par le serveur des calculs sur les objets de cette base. Ces services sont parfois récursifs : c'est à dire que le client peut à son tour publier le service auquel il est abonné et en devenir un des serveurs. Cette récursivité permet d'utiliser les Espaces de Travail pour construire des architectures de Gérant d'Objets ajustés aux environnements matériels sur lesquels ils fonctionnent.

Dans ce chapitre, nous réaliserons les architectures de Gérants d'Objets Persistants les plus diverses en imbriquant des Espaces de Travail entre eux. Nous décrirons d'abord les architectures simples qui représentent des modèles essentiellement adaptés à la micro-informatique. Les modélisations suivantes décriront des architectures Client-Serveur sur des hiérarchies de réseaux locaux ou longues distances.

L'Espace de Travail propose d'autres moyens d'accès aux données persistantes que le transfert des données vers l'application : le concepteur d'une base de données peut concevoir des services dédiés à ces applications. Ces services requièrent de la part du serveur une partie de calcul importante par rapport aux services de données. Les services d'Opérations proposent de faire calculer l'ensemble des opérations sur les objets par le serveur. Nous proposerons des services mixtes qui effectuent les calculs à la fois sur le client et sur le serveur. Ces services sont renforcés par une nouvelle définition des autorisations aux objets qui distingue si l'accès est réalisé par le serveur ou par le client.

Enfin les services coopératifs permettent la participation distante à un groupe de travail et à l'organisation du travail coopératif en plusieurs sous-groupe communicants. Nous conclurons enfin ce chapitre par un exemple d'architecture utilisant les différents services proposés.

II. Bases Privées

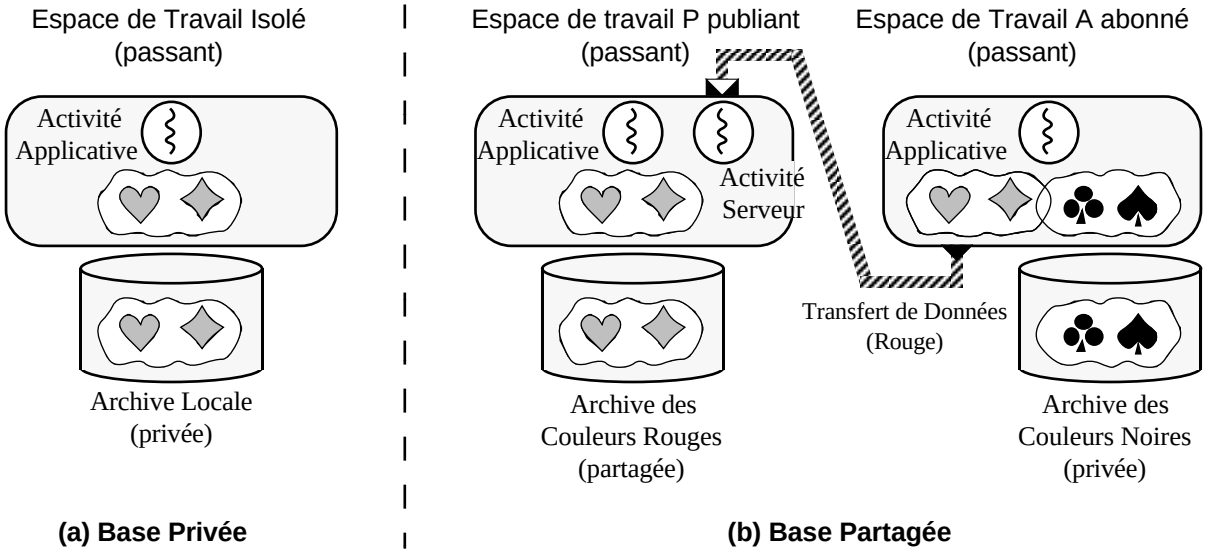
L'architecture de bases de données la plus simple est la base de données privée : elle correspond en général à une architecture matérielle qui se réduit à un ordinateur personnel isolé. Le succès de l'informatique personnelle a conduit les grands fournisseurs de systèmes bases de données à porter leurs produits vers cette informatique en plein essor. Dans cet environnement, l'utilisateur utilise seul une base de données privée archivée sur le disque de son micro-ordinateur.

L'Espace de Travail modélise directement cette architecture simple : il héberge l'application qui accède directement (au travers de l'espace de données) aux objets d'une archive locale. Une ou plusieurs activités effectuent les transactions de l'utilisateur (figure 4.1a). Dans ce contexte, l'Espace de Travail ne s'abonne pas à un service et n'en publie pas non plus.

La base privée regroupe l'application et l'archive dans le même Espace de Travail : l'application accède directement aux objets de l'archive sans subir les pénalités liées aux

communications ou pseudo-communications[1] des architectures Client-Serveur. La base privée dépasse le cadre de l'informatique personnelle : dans le cadre d'un atelier de génie logiciel, le concepteur d'une application doit pouvoir mettre au point son application sur une base de test sans mettre en oeuvre un serveur archivant la base de test. Dans le cas de logiciels embarqués, les données persistantes sont toujours privées; l'accès est ainsi optimisé par cette structuration.

Figure 4.1: Base Privée (a) et Bases Partagées (b)



III. Bases Partagées

Le développement des réseaux locaux a permis le partage des fichiers entre les différents postes de travail. Dans un tel environnement, les fichiers partagés ne sont pas toujours centralisés sur une machine qui joue le rôle de serveur. Le partage de fichier peut aussi émaner d'un utilisateur qui souhaite rendre public et accessible une partie de ses fichiers locaux. La station de l'utilisateur devient alors serveur de ces fichiers.

La publication de fichiers privés peut se transposer aux bases de données : un utilisateur propose un accès aux données de sa base privée. L'Espace de Travail, qui utilise cette base privée, publie le service de Données pour les objets de cette base; les autres Espaces de Travail peuvent s'abonner à ce service pour importer ces objets. Dans l'exemple de la figure 4.1.b, l'Espace de Travail P publie un service de Données sur les objets de son archive, l'archive des Couleurs Rouges. Un autre Espace de Travail A peut alors s'abonner auprès de P devenu serveur. Une transaction de service est instanciée pour dialoguer avec le nouveau client. L'Espace de Travail client peut néanmoins posséder sa base privée, l'archive des Couleurs Noires.

[1] Les pseudo-communications sont des communications entre deux processus s'exécutant sur la même machine : elles impliquent la copie du message délivré de l'espace d'adressage de l'émetteur vers celui du récepteur.

Cette structure possède cependant une faiblesse qui se retrouve dans toute la micro-informatique. Le serveur doit assurer une continuité de service auprès de ses clients. Or comme cet Espace de Travail pilote à la fois l'application d'un utilisateur et l'abonnement d'un client à un service, il est difficile de garantir la fiabilité des service proposé si l'application risque d'arrêter accidentellement l'Espace de Travail (crash). Dans cet exemple, une panne du serveur ne permettrait pas au client de valider les modifications faites sur des objets de l'archive distante de Couleurs Rouges. La solution à ce problème réside dans l'approche Client-Serveur dans laquelle la réalisation des services est isolée de toute application non fiable.

IV. Bases Client-Serveur

L'architecture Client-Serveur des Gérants d'Objets isole la gestion de la persistance des données de l'exécution de l'application. Le client est la tâche qui exécute l'application et le serveur centralise les données et en maintient une archive persistante. Le serveur est un Espace de Travail qui constitue son espace de données avec une archive locale et publie un service de Données sur cette archive. Le client est un Espace de Travail qui constitue son espace de données à partir d'un abonnement au serveur de données. Le client exécute une ou plusieurs transactions simultanées qui correspondent à une ou plusieurs applications travaillant pour le même utilisateur dans le cadre d'une méta-application (cf. chapitre 4 section IV.2.5).

Les tâches client et serveur peuvent être :

- sur la même machine physique,

  La séparation des tâches apporte la sécurité et la continuité du service de Données quand la base est accédée par plusieurs applications clients : les applications ne peuvent pas arrêter le serveur de données; et d'autre part, le serveur peut vérifier si l'application ne corrompt pas la base lors de la validation de modifications.

- sur des machines physiquement distinctes.

  Dans un environnement distribué, les communications sont les seuls moyens d'échange pour deux machines distinctes.

La base de données peut être répartie entre plusieurs archives physiquement stockées sur plusieurs machines serveurs. Nous proposons deux approches pour construire l'architecture Multi-Serveur :

- l'architecture **directe,**
- l'architecture **symétrique pair-vers-pair.**

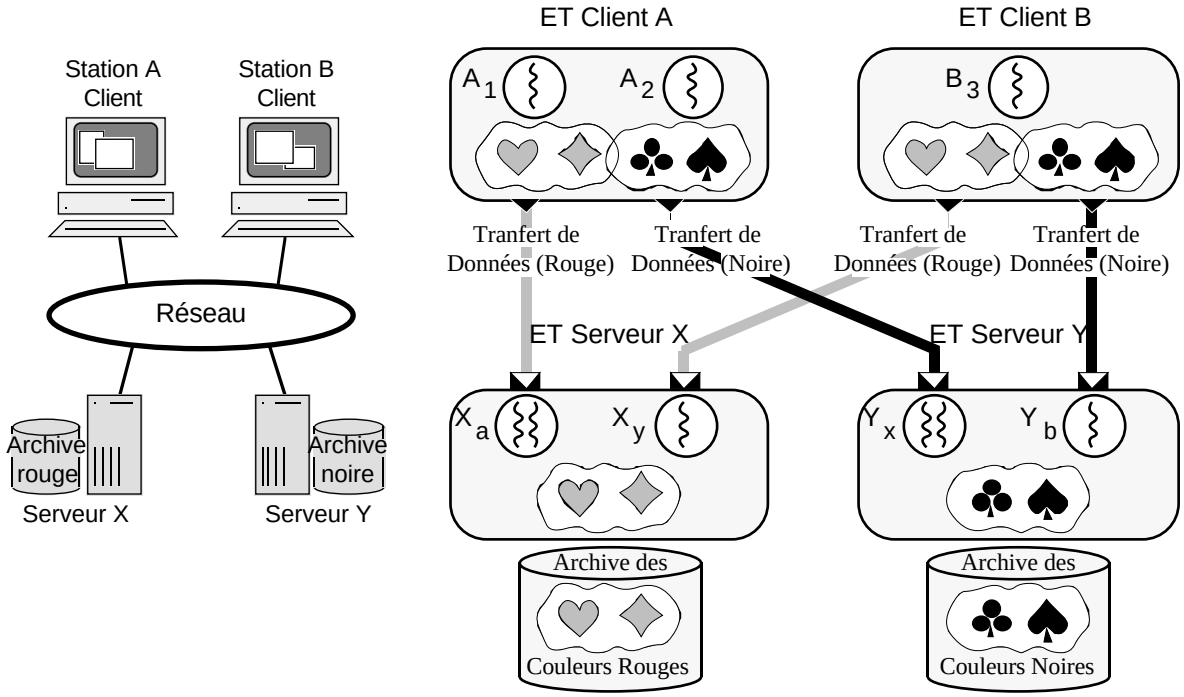
IV.1 Architecture Multi-Serveur Directe

Dans l'architecture directe, le client dialogue directement avec chaque serveur pour importer les portions de la base. Un Espace de Travail serveur est instancié sur chaque machine où réside une archive d'une portion de la base. Chaque Espace de Travail serveur publie le service de Données pour les objets de son archive locale : il est le seul serveur de ce service dans le

système. Chaque application (ou méta-application) est exécutée au sein d'un Espace de Travail client qui s'abonne aux différents services de Données des objets auxquels l'application désire accéder. L'espace de données du client constitue une vue globale de la base en fusionnant les vues apportées par les services de Données des différentes portions de la base.

Dans la construction de la figure 4.2, l'Espace de Travail A (comme B) s'abonne aux différents services de Données, publiées par les serveurs X et Y, pour accéder aux objets de l'archive rouge comme à ceux de l'archive noire. L'espace de données du client fusionne les deux vues apportées par ces services pour offrir une vue globale aux activités qui constituent l'application.

Figure 4.2: Bases Client-Serveur: l'approche directe.

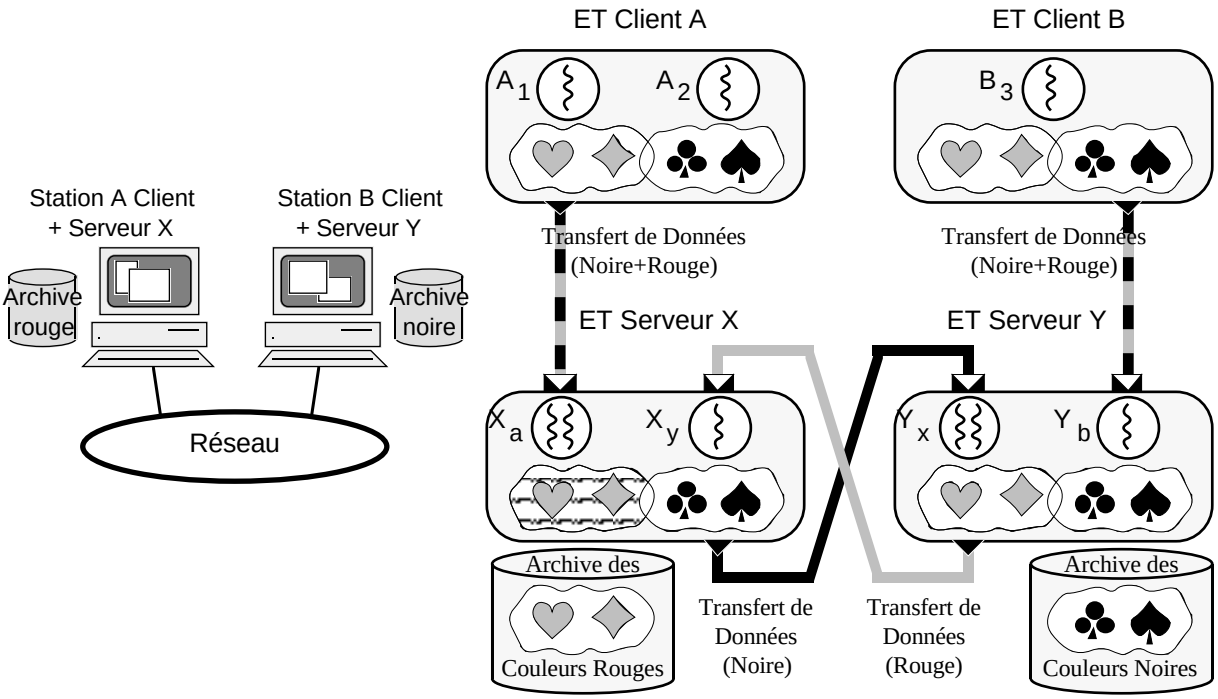


IV.2 Architecture Multi-Serveur Symétrique Pair-vers-Pair

L'architecture Symétrique Pair-vers-Pair (peer-to-peer symetric) propose de ne mettre le client en contact qu'avec un seul serveur. Ce serveur sert à ce client l'ensemble des objets de base. Ces objets sont soit stockés dans l'archive qu'il gère, soit sur des archive appartenant à d'autres serveurs. Dans ce dernier cas, le serveur se charge de dialoguer avec les autres serveurs du système pour obtenir les images des archives distantes et pour ensuite les "resservir" à son client.

Cette approche a été choisie par les concepteurs de SHORE [Carey94]. SHORE équipe toute machine du réseau d'un processus serveur avec lequel les applications clientes de la station dialoguent pour obtenir les données de la base entière. Le serveur masque ainsi au client la distribution de l'archive; il joue le rôle de concentrateur de données quand la station supporte plusieurs applications clientes.

Figure 4.3: Bases Client-Serveur: l'approche symétrique pair-vers-pair.



Dans la construction symétrique, l'Espace de Travail serveur publie deux services de Données :

- Le service Partiel propose une image de l'archive locale. Ce service est du même type que les services proposés par les serveurs de la construction directe. Cet Espace de Travail est le seul serveur de ce service dans le système. Cependant, contrairement à la construction directe, ce service est seulement destiné aux autres serveurs; ceux-ci l'utilisent pour définir le service Global.
- Le service Global propose une image globale de la base. Le serveur définit son espace de données à partir de son archive locale et à partir des abonnements aux services Partiels exportant l'image des autres archives. Ce service exporte l'image de son espace de données. Ce service, qui propose ainsi une image complète de la base, est publié par tous les serveurs du système; il est destiné aux applications clientes.

Le client s'abonne au service Global pour pouvoir importer n'importe quel objet de la base. Comme plusieurs serveurs publient ce service et que le client ne peut dialoguer avec qu'un seul d'entre eux (cf. chapitre 3 section III.1.2), le mécanisme de Publication-Abonnement résout ce conflit en choisissant le serveur le plus "proche" du client. Dans SHORE, c'est le serveur local de la station qui est choisi; dans ce cas, les échanges entre les clients et leur serveur sont des pseudo-communications.

Dans la figure 4.3, l'Espace de Travail A, qui exécute une application, s'abonne au service Global. L'abonnement est pris en charge par X qui se trouve être le serveur le plus proche pour servir l'union de l'archive de Couleurs Rouges et de celle des Couleurs Noires. Cet Espace de Travail X propose également le service Partiel sur l'archive de Couleurs Rouges; ce service est destiné aux autres serveurs pour concevoir le service Global. Réciproquement, X constitue le service Global à partir de son archive

locale des Couleurs Rouges et à partir du service Partiel des Couleurs Noires auquel il est abonné.

IV.3 Comparaison des Constructions Multi-Serveurs

IV.3.1. Contexte d'utilisation

Ces deux constructions Multi-Serveurs sont utilisées en fonction de l'implantation des applications exécutées par les clients :

- La construction directe considère l'Espace de Travail comme la seule application qui s'exécute sur la station de travail. Cette application correspond à la notion de méta-application que nous avons déjà évoqué (cf. chapitre 3 section III.2.5) : la méta-application regroupe plusieurs outils qui sont utilisés par l'utilisateur de la station pour la réalisation d'un seul travail. Les différents outils sont exécutés de manière concurrente par les activités de l'Espace de Travail client (toujours par le biais de transactions).

  Dans la figure 4.4 dans le cadre une application CASE, l'activité B_2 exécute un éditeur et l'activité B_3 un compilateur concurrent de B_2. Le client s'abonne aux services "Partiels" des serveurs X et Y pour importer directement leur archive; il joue le rôle de cache d'objets pour l'ensemble de ces activités.

- La construction symétrique suppose que plusieurs méta-applications s'effectuent sur la station, ou bien que les différents outils d'une méta-application doivent être isolés les uns des autres dans des clients différents. Le serveur local joue parfaitement son rôle de concentrateur des demandes d'objets par rapport au réseau. Ce rôle va de pair avec celui de cache des objets chargés par les clients de la station.

  Dans la figure 4.4, l'application CASE est conçue différemment pour exécuter l'éditeur et le compilateur dans des Espaces de Travail C et D séparées. Ces Espaces de Travail s'abonnent au service "Global" proposé par le serveur Y local à la machine M3. Ce serveur cache, pour l'ensemble des clients de la station, les objets importés des autres machines.

Les premières hypothèses d'implantation, supposées par la structure directe, pénalise l'architecture symétrique : celle-ci ajoute, sur la station, une tâche serveur qui ne fait que relayer les demandes de l'unique tâche client. Mis à part le surcoût du niveau de pseudo-communication entre le client et les serveurs des archives distantes, les objets sont cachés deux fois dans la mémoire de la station : une fois dans la mémoire du serveur local, une fois dans celle de l'application client.

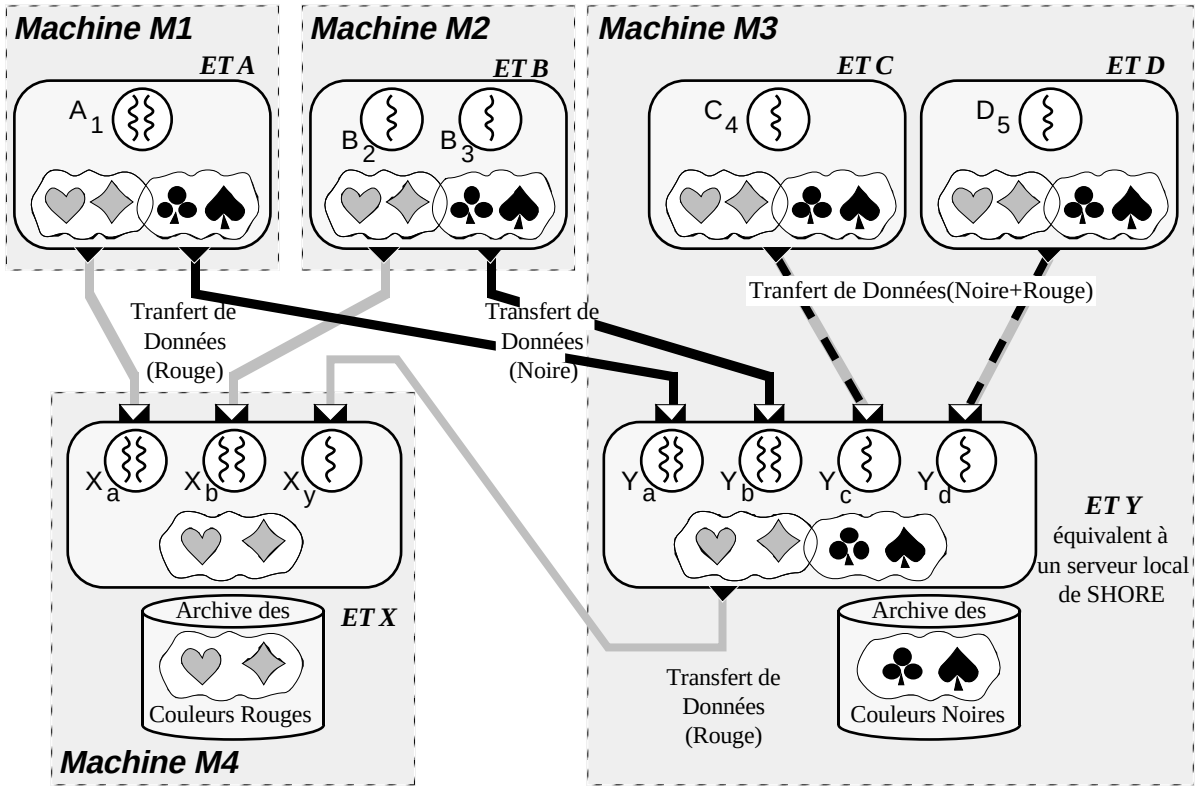
Inversement, si l'application, exécutée sur une machine, utilise plusieurs Espaces de Travail pour exécuter ses composants, la structure directe ne permet pas à ces Espaces de Travail clients de partager les communications pour importer le même objet : chaque Espace de Travail importe l'objet de son côté en dialoguant directement avec les serveurs.

Notons aussi que la construction directe suit la démarche de conception d'applications multi-sessions préconisée par les systèmes à espace d'adressage unique comme OPAL [Chase92]

exploitant l'adressage virtuelle des processeurs 64 bits ou bien celui du Newton d'Apple [Smith94] : en effet l'espace d'adressage unique évite les couteux changements de contexte sur le processeur.

Ces deux constructions restent compatibles entre elles; elles peuvent coexister dans la même architecture. Le choix d'une de ces deux constructions dépend de critère visant à optimiser l'utilisation des ressources locales d'une station. Le choix peut être décidé indépendamment pour chaque station : si une station n'exécute qu'un seul client et n'a pas d'archive à servir alors la construction directe peut être préférable à la construction symétrique. La figure 4.4 représente quatre machines sur un réseau : les applications, effectuées sur les machines M1 et M2, sont exécutées par des Espaces de Travail clients A et B qui importent directement les objets des différents serveurs X et Y. Par contre, les applications de la machine 3 sont exécutés par des Espaces de Travail clients C et D qui réclament localement les objets à l'Espace de Travail Y par le service Global. Ce serveur Y importe les objets externes par le même service que les clients A et B .

Figure 4.4 : Combinaison des deux structures multi-serveurs.



IV.3.2. Validation sécurisée des transactions

Les modifications apportées par un client, sont validées si et seulement si l'ensemble des serveurs valident correctement les modifications qui leur ont été adressées. Si un des serveurs ne valide pas sa part de modifications reçues, alors il faut abandonner l'ensemble des modifications. Ce principe inspire les protocoles de validations distribuées 2PC (Two-Phase Commit) [Özsu91] qui requièrent tous deux un site Coordinateur pour les différentes phases de la validation.

Dans le cas de la structure directe, le client est le seul coordinateur possible : en effet, aucun des serveurs ne sait avec quel serveur, le client est connecté. Le client est le coordinateur naturel du protocole : les serveurs sont donc esclaves du client et obéissent aveuglément aux ordres de pré-commit, de confirmation de commit ou d'avortement. Dans une telle situation, le client malintentionné peut enfreindre en forçant un serveur à valider tandis qu'un autre abandonne la validation. Cette situation pose réellement un problème dans le cas où les serveurs font subir aux modifications un contrôle de contraintes d'intégrité avant de les valider [Simon86]. Ces contraintes d'intégrité sont des règles, définies par le concepteur de l'application, qui vérifient si la base reste sémantiquement correcte après les modifications d'une transaction. Le calcul de ces contraintes nécessite de connaître la valeur des données modifiées (ce sont elles qui causent le changement d'état de la base) ainsi que des données non accédées par la transaction client.

La structure symétrique est la solution la plus abordable pour réaliser le contrôle des contraintes d'intégrité : le client importe les données et valide ces modifications au travers d'un seul serveur (qui est le serveur local si la machine en est munie). Au moment de la validation, ce serveur réalise le contrôle des contraintes d'intégrité sur les données renvoyées par le client, puis il coordonne le protocole de validation avec les autres serveurs si les modifications sont correctes. Ces derniers reçoivent du coordinateur les données modifiées qu'ils valident sans vérifier les contraintes d'intégrité sur celles-ci. Ce contrôle par le serveur coordinateur est une solution centralisée : le serveur peut devoir importer des données des autres serveurs pour calculer les contraintes (par exemple, supposons qu'une contrainte vérifie si la valeur modifiée d'un salaire ne dépasse pas 50% de la moyenne des salaires; le serveur qui vérifie cette contrainte, doit importer les données "salaires" présentes sur les autres serveurs afin de calculer la moyenne).

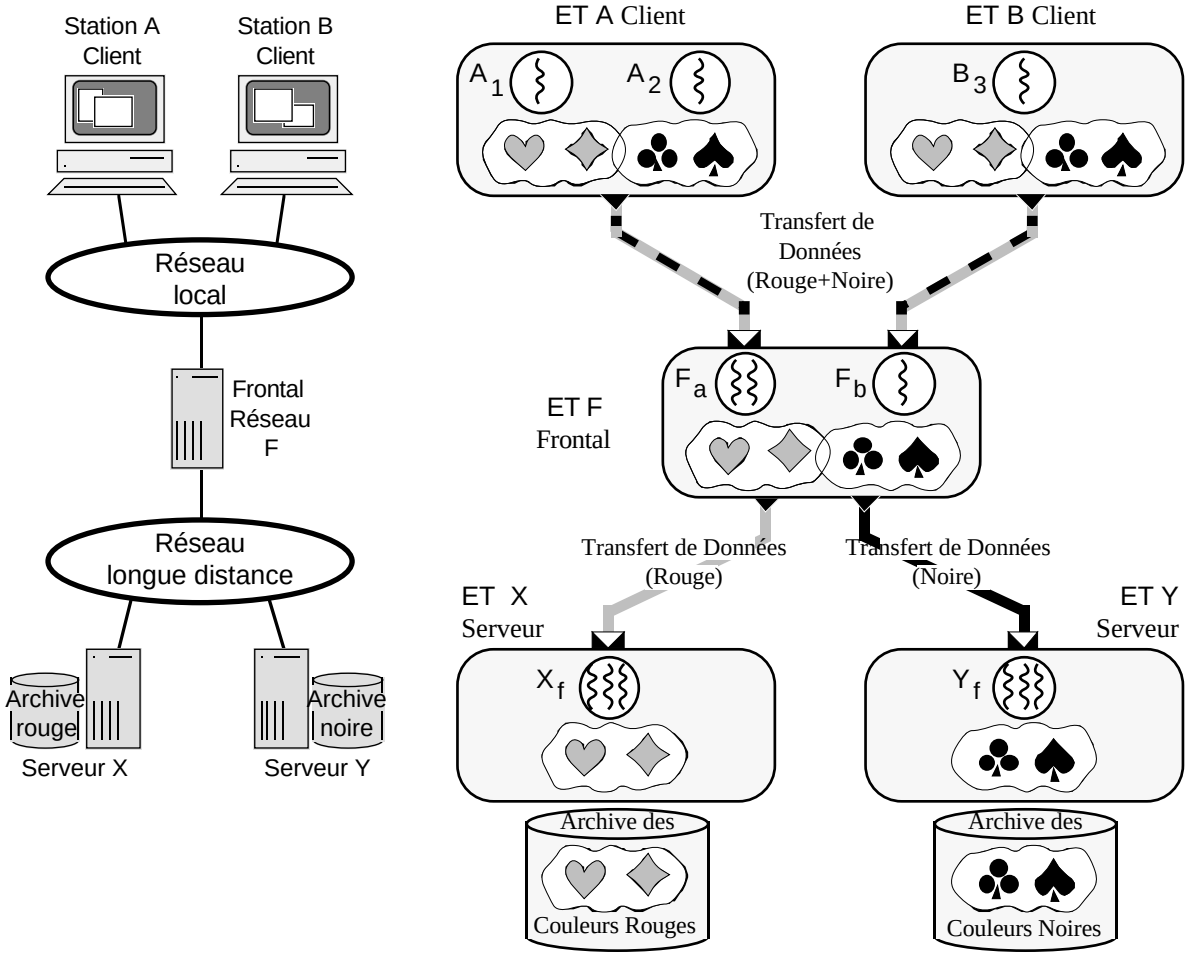
Comme un client peut effectuer des transactions simultanées, le serveur doit distinguer des modifications lors de la validation de ces transactions; l'activité de service héberge une transaction de service pour chaque transaction cliente; une fois que les données modifiées sont "redescendues" dans l'espace privé de la transaction de service, celle-ci exécute le programme de vérification des contraintes (qui peut nécessiter d'autres données), puis décide de valider ou d'abandonner les modifications.

Le serveur vérificateur-coordinateur peut être le serveur local si la station en est munie (cas de SHORE) ou un serveur distant sur une autre machine. Ce dernier cas est toujours utilisé quand la station du client n'est pas jugée digne de confiance : le serveur doit être sur une machine sécurisée. Les serveurs sécurisés distinguent donc les validations provenant d'un client non sécurisé, qui nécessite de vérifier des contraintes, et les validations provenant d'un autre serveur sécurisé.

V. Frontal Réseau Distant

L'informatique de l'entreprise se trouve souvent répartie entre plusieurs centres de traitement. Le centre de traitement héberge plusieurs postes de travail au sein d'un réseau local haut débit. Ce centre est en général pourvu d'un serveur départemental qui maintient l'archive des données communes du centre de traitement. L'information globale est accessible en dialoguant avec le serveur central de l'entreprise ou bien avec les serveurs départementaux des autres centres de traitement.

Figure 4.5: Espace de Travail intermédiaire: frontal d'un réseau distant



Les échanges entre les centres de traitement sont réalisés par des communications longues distances (i.e. Wide Area Network) qui représentent une part non négligeable du coût de l'exploitation du système. Dans ces conditions, il convient d'éviter que deux postes de travail, dans le même centre, engagent deux communications séparées pour rapatrier une donnée "outre-centre" identique. Une machine du centre doit devenir le frontal réseau : elle concentre les demandes des postes vers les serveurs externes au réseau local et cache les données rapportées pour l'ensemble des postes du centre. Cette tâche incombe naturellement au serveur départemental du centre[2].

Nous modélisons ce frontal réseau avec un Espace de Travail frontal relayant les transfert entre les clients du centre et les serveurs distants. Cet Espace de Travail s'abonne aux services de Données publiés par les différents serveurs; il publie ensuite au sein du réseau local, un service de Données qui fusionne l'image des différentes archives distantes. La station cliente est un Espace de Travail qui s'abonne à ce service de Données pour accéder aux objets externes au réseau. Dans le réseau local, le frontal apparaît comme un serveur quelconque et l'architecture

2 La machine, choisie pour remplir ce rôle, est souvent la passerelle physique des communications entre le réseau local et le réseau longue distance.

du gérant d'objets, à l'intérieur de ce réseau, est réalisée avec les constructions Multi-Serveurs de la section III.

Dans la figure 4.5, le frontal F s'abonne aux services de Données des serveurs X et Y par des communications longue distance; il publie sur le réseau local un service de Données qui fusionne les services auxquels il s'est abonné. Les clients A et B peuvent s'abonner à ce service pour avoir une image globale des archives distantes. Le frontal F joue de rôle de cache pour les clients; si un des clients demande une donnée; celle-ci est demandée par F au serveur sur lequel elle est archivée; la donnée est alors cachée pour les demandes suivantes.

Le mode de communication entre le frontal et les serveurs centraux est indépendant de l'abonnement et peut être adapté à la nature du réseau de communication. Par exemple, si le réseau longue distance utilisé est le Réseau Téléphonique Commuté dont le coût de communication dépend de la durée de connexion, le frontal pourrait décider de se déconnecter du serveur pour optimiser le coût sans pourtant se désabonner du service. Ces considérations se retrouvent aussi en informatique mobile [Imielinski92, Barbara94].

VI. Serveurs d'Opérations

L'architecture des Gérants d'Objets est construite sur le Migration des Données qui est offert par l'Espace de Travail via le service de Données. Cependant la Migration des Requêtes reste encore une nécessité pour les applications sécurisées ou pour des opérations peu sélectives. Or ces gérants d'objets ne permettent pas de réaliser, de façon efficace, ce type de service : en général, c'est un client du serveur de données qui propose ce service (chapitre 3 section II.3.1). L'Espace de Travail unifie la notion de client et de serveur : il permet de servir des données qu'il archive (service de Données), et effectuer des calculs sur ces données pour le compte des plusieurs clients (service d'Opérations).

Nous proposons dans cette section deux constructions d'Espaces de Travail offrant les services d'Opérations. La première correspond au problème du serveur implanté au niveau client; la seconde réalise les services directement sur l'Espace de Travail serveur de données. Nous terminons par les spécificités des tâches abonnées aux services d'Opérations.

VI.1. Serveur implanté au niveau du client

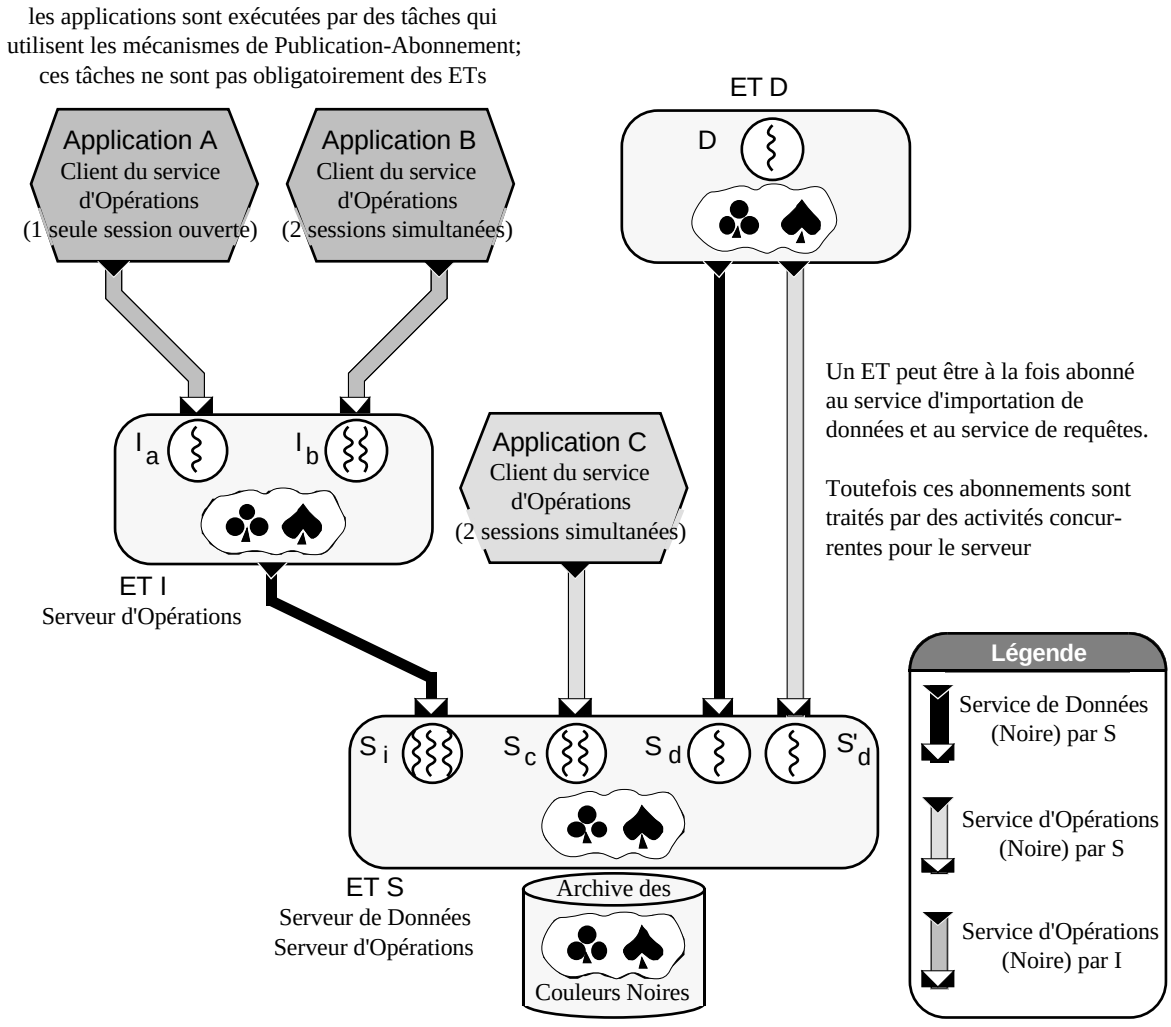
La première construction reprend l'implantation actuelle des services de Migration de Requêtes. Rappelons que dans les Gérants d'Objets actuels, le serveur n'offre pas d'espaces privés de modification pour y exécuter des transactions. Dans ce contexte, l'espace privé de modifications correspond au client de ce serveur. Le client peut donc proposer des services qui entraînent des modifications sur ce client devenu serveur. Ces modifications sont réalisées par des transactions de ce client.

La première construction repose donc sur un Espace de Travail intermédiaire entre l'Espace de Travail serveur de données et les clients du service d'Opérations. Cet Espace de Travail s'abonne au service de Données pour accéder aux objets de la base au travers de l'espace de données. Le concepteur du service définit l'ensemble des opérations qui peuvent être réalisées

dans le service d'Opérations que publie l'Espace de Travail. Les abonnements des clients sont pris en compte par des activités qui instancient des transactions pour calculer les requêtes des clients : les transactions de service consultent et modifient les objets de l'espace de données.

Dans la figure 4.6, l'Espace de Travail I remplit le rôle d'intermédiaire : il s'abonne au service de Données, publié par le serveur de données S, puis publie le service d'Opérations pour les applications clients A et B. Il instancie une activité par abonnement; chaque activité exécute une transaction pour calculer les requêtes de A et B en accédant aux objets de l'espace de données de I.

Figure 4.6 : Hiérarchie de Serveurs d'Opérations.



VI.2. Serveur multi-services

La deuxième construction exploite directement les propriétés de l'Espace de Travail : l'Espace de Travail banalise la notion d'application ou de services en faisant réaliser les deux par des transactions qui accèdent aux objets de l'espace de données. Le serveur de données exécute aussi bien les transactions d'une application comme celles associées aux abonnements d'un service d'Opérations.

Le serveur S de la figure 4.6 publie à la fois le service de Données (pour I et D) et le service d'Opérations déjà publié par S (pour C et D). La réalisation des requêtes ne nécessite pas de déplacement d'objets lors du calcul de la requête.

VI.3. Spécificités des clients d'un service d'Opérations

Le service d'Opération est destiné à des clients qui en principe n'ont pas besoin d'espace de données comme l'Espace de Travail (comme les tâches A, B, C dans la figure 4.6). Chacune de ces clients ouvre un ou plusieurs sessions indépendantes; une session est une suite de requêtes, envoyées au serveur, dont les modifications sont validées en fin de session (cf. chapitre 3 section IV.3.1). Chaque session est exécutée par une transaction indépendante de l'activité de service. Dans la figure 4.6, les tâches clientes B et C ouvrent chacune deux sessions simultanées; les activités de services I_b et S_c correspondantes instancient une transaction indépendante pour chaque session ouverte sur leur client.

Un Espace de Travail, déjà client d'un service de Données, peut aussi requérir à un service d'Opérations pour accéder à une base d'objets. Ce type de double abonnements est conflictuel quand les deux services accèdent à la même base de données. En effet, ces deux abonnements sont réalisés par deux transactions qui sont concurrentes lors des accès aux objets. Dans la figure 4.6, l'Espace de Travail D s'abonne au service de Données et au service d'Opérations publiés par le serveur S. Les activités S_d et S'_d, associés aux abonnements, sont en conflit lors des consultations et des modifications des objets de l'espace de données. La section suivante résout ce problème en proposant un service qui fusionne les deux services.

VII. Service Mixte de Données et d'Opérations

Nous proposons dans cette section d'effectuer des opérations à la fois sur le Client et sur l'Espace de Travail serveur. Ceci fait l'objet du service Mixte qui fusionne le service de Données et un service dédié de type Opération. Une des principales fonctions du service Mixte est le maintien de la cohérence de l'espace de données du client et l'espace privé de la transaction qui réalise le service sur le serveur.

Les services Mixtes s'appliquent dans les environnements sécurisés ou pour le calcul de transactions distribuées. Dans un environnement sécurisé, les services Mixtes s'accompagnent d'une extension des autorisations d'accès usuelles qui est décrite à la section VIII. Les autorisations d'accès à un objet sont différenciées selon que l'opération soit effectuée par le client ou bien par le serveur sécurisé.

VII.1. Contexte

Les architectures, présentées dans les sections III à VI, s'appuient soit sur le service de Données, soit sur un service d'Opérations. Le service de Données s'adresse généralement à des Espaces de Travail, alors que les services d'Opérations sont destinés principalement aux autres tâches du système qui n'ont pas besoin d'un espace de données comme l'Espace de Travail.

Dans les contextes sécurisés, le client n'est pas toujours jugé digne de confiance pour exécuter certaines opérations ou consulter certaines données : une partie des opérations de l'application doivent être effectuées sur un site sécurisé. Ces opérations sont donc regroupées dans un service publié par un serveur sécurisé. Toutefois l'application du client doit pouvoir accéder directement aux objets pour lesquelles elle possède des autorisations d'accès suffisantes. Cet accès se fait par le service de Données proposé par le même serveur.

VII.2. Fusion des Services

Les abonnements au service de Données et au service d'Opérations sont réalisés par deux transactions distinctes de l'Espace de Travail serveur. Ces transactions qui accèdent au même ensemble d'objets, sont concurrentes dans les consultations et les modifications de ces objets bien qu'elles travaillent toutes deux pour le même client.

Nous proposons de résoudre ce conflit d'accès aux objets en fusionnant les deux services en un seul. Le service Mixte regroupe à la fois :

- les opérations d'échanges d'objets proposées par le service de Données,
- les opérations de calcul d'un service d'Opérations, spécifiques à l'application et définies par le concepteur du service.

L'abonnement au service Mixte est réalisé de manière identique aux autres services : le serveur laisse une de ses activités dialoguer avec le client; toutefois l'activité de service n'instancie qu'une seule transaction à la fois.

VII.3. Maintien de la Cohérence entre les données du Client et celles de l'Activité de Service

Dans le service Mixte, les objets sont modifiés à la fois dans l'espace de données du client, et dans l'espace privé de la transaction associée à l'activité de service. Le démarrage d'un calcul dans un espace nécessite que cet espace soit cohérent par rapport à l'autre, c'est à dire qu'il contient la valeur à jour des objets. Quand l'application effectue des calculs qui modifient les objets d'un espace, les valeurs de ces objets deviennent périmées dans l'autre espace. Les calculs sont réalisés :

- soit directement par le client,

  L'espace de données du client contient de nouvelles valeurs des objets; l'espace privé de la transaction de service contient des valeurs périmées de ces objets.

- soit par la transaction de service lors d'une requête au service dédié.

  L'espace privé de la transaction de service contient de nouvelles valeurs des objets; l'espace de données du client contient des valeurs périmées de ces objets

Ces deux espaces doivent être cohérents quand le client ou la transaction de service accèdent à ceux-ci. Cette cohérence n'est pas nécessairement totale sur l'espace entier car les accès peuvent ne concerner qu'une partie des objets dont les valeurs sont périmées.

Notre approche consiste à mettre à jour la valeur périmée d'un objet au moment de l'accès à cette valeur. L'accès est détecté par le mécanisme de défaut d'objets augmenté de la possibilité d'invalider l'image d'un objet dans un des 2 espaces. Le mécanisme de défaut d'objets est déjà utilisé par le service de Données et nous verrons dans la section III du chapitre 5, les détails de l'implantation de ce mécanisme. Nous allons détailler les modalités d'utilisation des invalidations. L'invalidation d'un objet a lieu dans un des espaces chaque fois que la valeur de l'objet devient périmée. L'invalidation a lieu quand :

- l'objet est modifié dans l'espace client.

  l'objet doit être invalidé dans l'espace privé de la transaction serveur; le client envoie une demande d'invalidation à la transaction serveur[3]. Si la transaction serveur a besoin ultérieurement de l'objet, le défaut d'objet provoque la demande de "redescente" de l'objet par la transaction serveur.

- l'objet est modifié dans l'espace privé de la transaction serveur et il existe sur le client une image cohérente de l'objet.

  Le serveur envoie la demande d'invalidation de l'objet au client[4]. Par la suite, après le retour de l'opération, si le client accède à l'objet, le défaut d'objet provoque la "remontée" de l'objet vers le client.

Illustrons ces mises à jour d'espaces de données par l'exemple des transactions bancaires. Un utilisateur désire lors d'une transaction (a) consulter le montant de son compte bancaire, (b) changer sa domiciliation, (c) transférer une somme depuis son compte vers celui d'un autre et enfin (d) consulter le nouveau montant. Les opérations (a), (b), (d) sont réalisées par le client, mais les contraintes de confidentialité et de sécurité impose la réalisation du transfert de fond (c) par le serveur sécurisé. Le client exécute la transaction de la figure 4.7 :

(a) Le client consulte l'objet `monCpt`.

  Cette consultation provoque un défaut de l'objet dans l'espace du client. (1) Le client demande alors l'image de l'objet. La transaction serveur consulte l'objet ce qui provoque un défaut d'objet dans l'espace privé de celle-ci. L'objet est récupéré dans l'espace de données puis (2) il est expédié au client.

(b) Le client consulte l'objet `monAdr` puis le modifie.

  La consultation requiert les mêmes actions (3) (4) mais la modification de l'objet `monAdr` provoque (5) l'envoi de l'invalidation de cet objet dans l'espace privé de la transaction serveur. Une consultation ultérieure de cet objet par la transaction serveur provoquerait un défaut d'objet dans l'espace privé de celle-ci. L'objet devrait alors être redescendu.

---

3 La première invalidation demandée correspond généralement aussi la demande de verrouillage en écriture de l'objet. Les invalidations suivantes peuvent être bufferisées avant d'être envoyées au serveur, toutefois une demande d'opération purge ce buffer.

4 Comme dans le cas précédent, les invalidations sont bufferisées avant d'être envoyées au client, jusqu'au retour de la demande d'opération qui purge ce buffer.

(c) Le client demande à la transaction serveur de réaliser le calcul correspondant au transfert de fond.

(6) La demande est envoyée à la transaction serveur qui exécute les lignes L3a et L3b de la méthode de transfert. La ligne L3a provoque un défaut de l'objet `sonCpt` dans l'espace privé de la transaction serveur. La ligne L3b modifie l'objet `monCpt`, et (7) provoque ainsi l'invalidation de cet objet dans l'espace du client. (8) La fin de la requête (accompagné du résultat) est signalée au client.

(d) Le client consulte de nouveau l'objet `monCpt`.

L'objet `monCpt` a été invalidé (7) alors la consultation provoque un défaut d'objet. (9) Le client demande la nouvelle valeur de cet objet à la transaction de service qui lui renvoie(10), puis consulte l'objet.

La nature du client d'un service Mixte est intermédiaire entre l'Espace de Travail qui s'abonne au service de Données et la tâche quelconque du système qui s'abonne à un service d'Opérations. Le client du service Mixte doit posséder les trois caractéristique suivant :

- avoir un espace de données semblable à celui de l'Espace de Travail : nous utilisons les mêmes mécanismes pour implanter un tel client.
- n'utiliser qu'une seule transaction à la fois et utiliser l'espace de données comme espace privé de modification de cette transaction.
- n'effectuer que des appels de requêtes synchrones (section III.1.1 du chapitre 3) : en effet, les exécutions du client et du serveur doivent se synchroniser sur les appels et les retours des requêtes de calcul pour éviter que l'un ou l'autre ne consulte des objets qui sont sur le point d'être invalidés.

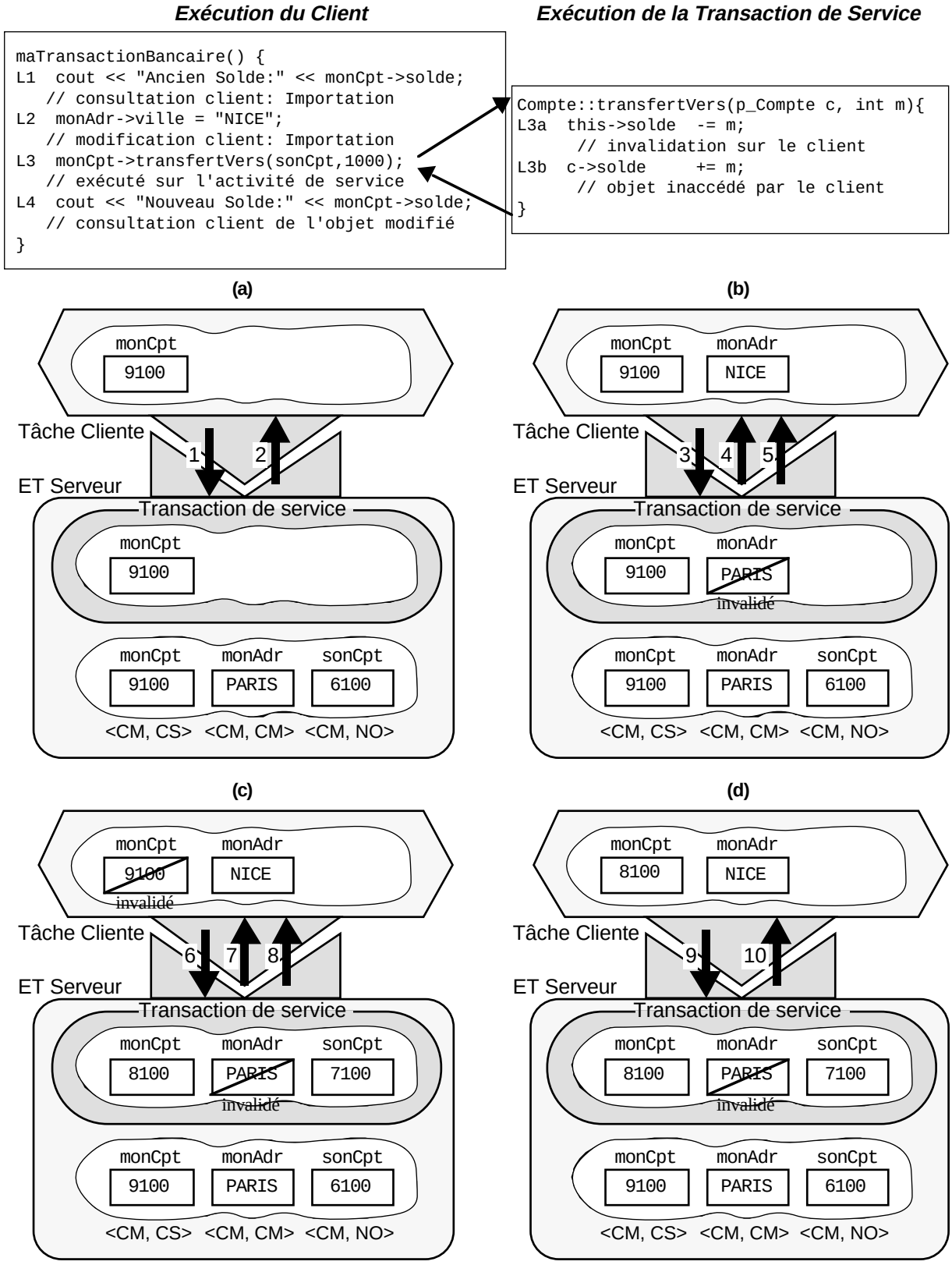
VII.4. Validation des Modifications

Les modifications, apportées par l'application, sont partiellement présentes dans l'espace de données du client et partiellement présentes dans l'espace privé de la transaction serveur. La validation des modifications, qui est décidée par le client, doit rendre visible l'union de ces modifications dans l'espace de données du serveur.

La validation implique la redescente d'une partie des objets, modifiés par le client, vers la transaction serveur. Les objets redescendus sont les objets modifiés dans l'espace du client et invalidés dans l'espace privé de la transaction serveur. Dans l'exemple de la figure 4.7, si la validation intervient après la transaction bancaire (après (d)), l'espace du client contient deux objets modifiés. L'objet `monCpt` n'est pas redescendu puisque sa valeur est à jour dans l'espace privé de la transaction serveur. Par contre, l'objet `monAdr` est redescendu vers le serveur puisque sa valeur, présente dans l'espace privé, est invalidée.

Une fois que son espace privé a été rendu cohérent par les redescentes, la transaction serveur peut ensuite être validée comme une transaction quelconque en modifiant dans l'espace de données du serveur les nouvelles valeurs des objets modifiés. Dans l'exemple de la figure 4.7.d, les trois objets `monCpt`, `monAdr`, et `sonCpt` sont validées par la transaction serveur comme le ferait une autre transaction applicative de l'Espace de Travail serveur.

Figure 4.7 : Echanges du service Mixte dans un système de transaction bancaire.



VIII. Services et Autorisations d'accès

VIII.1. Introduction

Le concept des autorisations d'accès est indispensable dans un système dans lequel travaillent les applications de plusieurs utilisateurs [Garfinkel91]. Elles définissent les limites lors des accès aux ressources du système suivant l'identité de la tâche qui y accède. La tâche prend l'identité d'un utilisateur pour lequel elle travaille. Dans un contexte de service, le serveur, qui a ses propres autorisations d'accès, exécute les requêtes du client en tenant compte de l'identité de son client.

Dans les systèmes usuels, les autorisations d'accès sont des variables à deux dimensions affectées aux ressources. La première dimension définit la nature des opérations autorisées[5] et la seconde dimension concerne l'identité de l'utilisateur qui accède à la ressource[6] [Mullender89 chapitre 7, Goscinski91 chapitre 10]. Dans un système Bases de Données, les objets sont les ressources sur lesquelles les accès sont contrôlés.

Dans cette section, nous proposons d'étendre le concept usuel des autorisations d'accès pour accompagner les services proposés par les Espaces de Travail. Cette extension distingue pour un utilisateur les modes d'opérations autorisées sur un objet quand l'opération est effectuée par le client et quand elle est effectuée par le serveur.

Nous proposons également de propager ces autorisations dans le graphe formé par les abonnements récursifs. Lors de l'abonnement à un service, les autorisations sont propagées vers le client. Ce dernier peut proposer à son tour des services et propager de nouvelles autorisations d'accès vers ses clients. Cependant, ces nouvelles autorisations ne peuvent pas dépasser celles contrôlées par le serveur sur le premier client.

VIII.2. Autorisations d'Accès aux Objets.

Dans le modèle des Espaces de Travail, les ressources dont il faut contrôler les accès sont les objets. Dans l'espace de données d'un Espace de Travail, les objets sont accédés soit par des transactions d'une activité applicative, soit par des transactions d'une activité de service. L'Espace de Travail définit des autorisations d'accès pour chaque objet; les transactions doivent vérifient celles-ci avant d'accéder à l'objet.

Nous définissons une relation d'ordre R entre les modes d'opérations possibles. Pour la clarté de l'exposé, nous nous limitons aux opérations classiques en Bases de Données.

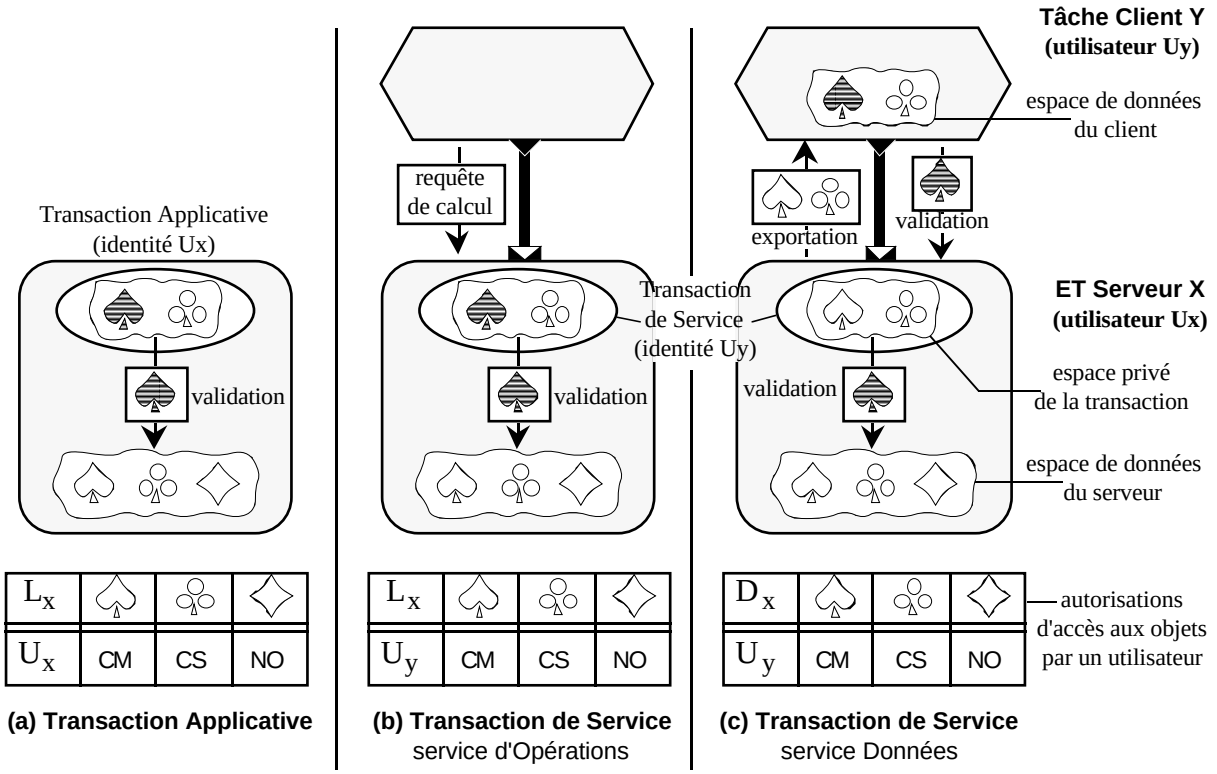
---

5 Sur VMS, les modes d'opération sont la consultation, l'exécution, la modification, la destruction et la création de fichier. Unix regroupe ces trois derniers dans un seul mode, le mode Ecriture.

6 Les systèmes distinguent en général les 3 types suivants d'utilisateurs :
- le propriétaire à qui appartient la ressource et qui, à ce titre, administre celle-ci comme bon lui semble,
- un groupe d'utilisateurs privilegiés et
- les autres.

Sur Unix, le groupe est une structure assez rigide qui est définie dans l'ensemble du système par son administrateur, alors que Multics laisse le propriétaire d'une ressource intégrer n'importe quel utilisateur dans le groupe de la ressource

Figure 4.8 : Autorisations d'accès pour des Applications et des Services.



R : **Pas d'Opération (NO) < Consultation Seule (CS) < Consultation+Modification (CM)**

Les autorisations d'accès à un objet se définissent de la manière suivante. L'Espace de Travail X définit les deux variables d'autorisation suivantes pour chaque objet O de son espace de données D_X et pour chaque utilisateur U du système :

- **L_X (O,U) la variable d'autorisation locale,**

  la variable d'autorisation locale définit le mode d'accès le plus élevé des opérations sur l'objet O par rapport à la relation d'ordre R quand l'opération est exécutée localement sur le serveur par une transaction attachée à l'utilisateur U.

- **D_X (O,U) la variable d'autorisation distante,**

  la variable d'autorisation distante définit le mode d'accès le plus élevé des opérations sur l'objet O par rapport à la relation d'ordre R quand l'opération est exécutée sur le client dont l'identité est celle de l'utilisateur U. Cette variable est utilisée par la transaction du service de Données pour contrôler les exportations d'objets et les validations d'objets modifiés.

VIII.2.1. Autorisations d'une Transaction Applicative.

Une transaction applicative utilise l'identité de l'utilisateur U_X de l'Espace de Travail X sur laquelle elle est exécutée. Lors de l'exécution d'une opération, elle vérifie si cette opération ne dépasse pas les autorisations d'accès définies par la variable d'autorisation $\mathbf{L_X\ (O,U_X)}$.

> D'après les autorisations définies dans la figure 4.8.a, une transaction applicative peut consulter et modifier l'objet Pique et seulement consulter l'objet Trèfle. Mais aucune opération ne lui est permise sur l'objet Carreau.

Quand l'application s'exécute sur un Espace de Travail qui gère une archive locale, ces autorisations ne correspondent pas à une protection véritable contre la malveillance : les transactions de cet Espace de Travail ont la possibilité de réaliser n'importe quelle opération sur ces objets. Cependant dans le contexte de service, les transactions de services sont censées vérifier scrupuleusement les autorisations définies pour leur client lors des accès aux objets.

VIII.2.2. Autorisations d'une Transaction de Service.

Quand une transaction répond à une demande de service auquel s'est abonné un client Y, elle utilise l'identité de l'utilisateur U_Y du client Y pour vérifier les autorisations d'accès. Dans le cadre d'un service quelconque, la transaction de service reçoit deux types de requêtes provenant du client:

- les requêtes d'Opérations

  Le client demande à la transaction de service d'effectuer un calcul consultant et/ou modifiant un ou plusieurs objets.

- les requêtes de Données

  Le client demande à la transaction de service de lui expédier l'image brute d'un objet (exportation) ou de mettre à jour (validation) l'espace de données de l'Espace de Travail serveur avec la valeur de l'objet modifié par le client.

La transaction de service effectue un contrôle différent dans ces deux cas :

- Lors d'une requête d'Opérations, elle vérifie avant chaque opération du calcul sur un objet O si l'opération n'excède pas la variable d'autorisation locale $\mathbf{L_X\ (O,U_Y)}$ définie pour l'utilisateur U_Y.

  > D'après les autorisations définies dans la figure 4.8.b, la transaction de service peut consulter et modifier l'objet Pique et seulement consulter l'objet Trèfle pour réaliser des calculs localement au serveur. Aucune opération ne lui est permise sur l'objet Carreau.

- Lors d'une requête de Données, elle vérifie en fonction la variable d'autorisation distante $\mathbf{D_X\ (O,U_Y)}$ si l'objet O peut être expédié vers le client pour y être consulté ou modifié. Ainsi dans le cadre des trois modes d'opération { NO, CS, CM }, cette variable implique qu'une transaction de service vérifie :

- si D_X (O,U_y) est différente de NO avant d'exporter l'objet O vers le client d'identité U_y,
- si D_X (O,U_y) est égale à CM avant de valider la nouvelle valeur de l'objet O redescendue par le client.

D'après les autorisations définies dans la figure 4.8.c, la transaction de service peut exporter l'objet Pique et le valider s'il est modifié. L'objet Trèfle peut être exporté vers le client pour y être consulté seulement : la transaction de service n'accepte pas de valider la valeur modifiée de l'objet si celle-ci est redescendue. Enfin, l'objet Carreau ne doit pas être exporté vers le client.

VIII.2.3. Autorisations pour un Service Mixte.

Le service Mixte d'Opérations et de Données (cf. section VI) se place dans le cas précédent de contrôle. Une transaction répondant à un abonnement reçoit des requêtes de type Opérations et des requêtes de type Données. Dans un tel contexte, la configuration des deux variables L_X (O,U_y) et D_X (O,U_y) permet, au concepteur du service, de forcer un client à faire réaliser par le serveur certaines opérations sur les objets pour lesquelles le client n'a pas d'autorisation d'accès. Le tableau 4.9 dresse cette liste de requêtes utilisables dans le cadre d'un service Mixte en fonction de la valeur de ces deux variables.

Reprenons l'exemple de la transaction bancaire du service Mixte (figure 4.7 de la section VII). Dans cette exemple, les seules opérations réalisables par le client sont :

- la consultation et la modification de l'objet Adresse dont il est "propriétaire",
- la consultation seule de l'objet Compte dont il est "propriétaire".

Les objets Compte et Adresse, qui ne lui appartenient pas, ne peuvent être consultés pour des raisons de confidentialité, et les modifications ne peuvent être réalisées par le client de façon sûre : ces opérations doivent être effectuées par la transaction de service.

Le concepteur de l'application positionne les autorisations suivantes pour les objets Compte et Adresse :

Utilisateur \ *Objet*	Compte	*Autorisation* L_X	D_X
Propriétaire		CM	CS
Autres		CM	NO

Utilisateur \ *Objet*	Adresse	*Autorisation* L_X	D_X
Propriétaire		CM	CM
Autres		NO	NO

Ces autorisations ne permettent pas à un utilisateur autre que le propriétaire de consulter un objet Adresse.

Cette différenciation entre les autorisations d'accès du client et celles du serveur s'apparente beaucoup au "set-uid" bit du système Unix. Unix autorise les utilisateurs à accéder à des fichiers pour lesquels ils n'ont a priori aucun droit, à travers un programme marqué de ce set-uid bit. La transaction de service se comporte un peu comme la commande marquée du set-uid bit, en n'exécutant que des opérations enregistrées et définies par le concepteur du service.

VIII.3. Autorisations et Graphes de Services.

Les abonnements aux différents services entre les Espaces de Travail et les autres tâches forment un graphe. Nous proposons de propager les autorisations d'accès définies sur le serveur au travers des abonnements de ce graphe. Cette propagation peut augmenter ou diminuer les autorisations d'un utilisateur sur un objet. Nous proposons une règle de propagation des autorisations évitant d'introduire des failles dans le système d'autorisations.

VIII.3.1 Autorisation Maximale

Le client d'un service peut à son tour définir des autorisations sur les objets présents dans son espace de données et publier ensuite un service qui vérifiera ces autorisations lors de l'abonnement d'un client à ce service. Cette redéfinition des autorisations par cet Espace de Travail intermédiaire subit toutefois la contrainte suivante : les autorisations, redéfinies pour l'objet O pour n'importe quel utilisateur, ne peuvent dépasser la valeur de la variable d'autorisation distante D_X définie par le serveur X pour l'utilisateur U_Y du client Y.

La variable Maximale M_Y (O) (= D_X (O,U_Y)) de l'Espace de Travail client Y définit donc le seuil de mode d'opération qui ne doit être dépassé par les activités du client Y. Si une activité de Y enfreint ce seuil M_Y (O), le serveur X est conduit à refuser la réalisation d'une requête de données.

VIII.3.2 Exportation des Autorisations

Le client peut publier à son tour des services d'Opérations ou de Données basés sur les objets de son espace de données. Les transactions de service utilisent les variables d'autorisation suivantes pour contrôler les accès des clients Z (d'identité U_Z) :

- L_Y (O,U_Z) pour les requêtes de type Opération,
- D_Y (O,U_Z) pour les requêtes de type Donnée.

Figure 4.9 : Table des requêtes utilisables en fonction des variables d'autorisation et d'échange d'un objet.

$D_{X(O, U_y)}$ / $L_{X(O, U_y)}$	Aucune • pas d'exportation	Consultation Seulement • exportation • pas de "redescente"	Consultation et Modification • exportation • "redescente"
Aucune • pas d'opération	aucune requête	la variable Lx n'est jamais inférieure à la variable Dx	
Consultation Seulement • pas de modification	requête d'Opérations (consultation)	requête d'Opérations (consultation) requête de Données (consultation)	
Consultation et Modification	requête d'Opérations (consultation+ modification)	requête d'Opérations (consultation+ modification) requête de Données (consultation)	requête d'Opérations (consultation+modification) requête de Données (consultation+modification)

$$L_y (O,U_Z) \leq M_y (O) \text{ et } D_y (O,U_Z) \leq M_y (O) \quad \forall U_Z \ \forall O$$

La variable Maximale M_y (O) borne ces autorisations pour éviter que le serveur X ne refuse les requêtes de données émanant de ces activités de service de Y.

La redéfinition des variables L_y (O,U) et D_y (O,U) permet à l'utilisateur Uy, lors de la proposition de service par l'Espace de Travail Y, soit de diminuer les autorisations d'un utilisateur U_Z, soit d'augmenter celles-ci par rapport à celles définies sur le serveur X :

- La diminution des autorisations de l'utilisateur U_Z sur un objet O se caractérise par la relation suivante :

  $$L_y (O,U_Z) < L_X (O,U_Z) \text{ et } D_y (O,U_Z) < D_X (O,U_Z)$$

  Une entité d'identité U_Z, qui s'abonne au service proposé par Y, possède moins d'autorisation que s'il s'abonnait à un service proposé par X.

- L'augmentation des autorisations de l'utilisateur U_Z sur un objet O, se caractérise par la relation inverse :

  $$L_X (O,U_Z) < L_y (O,U_Z) \ (\leq M_y (O)) \text{ et } D_X (O,U_Z) < D_y (O,U_Z) \ (\leq M_y (O))$$

  Une entité d'identité U_Z, qui s'abonne au service proposé par Y, possède plus d'autorisations que s'il s'abonnait directement au service proposé par X. Par cette redéfinition, l'utilisateur U_y transmet une partie de ces "privilèges" à l'utilisateur U_Z. L'augmentation ne doit pas dépasser la variable Maximun M_y (O).

  Cette augmentation contourne les protections mises en place, par l'utilisateur U_X du serveur X, pour limiter les droits de l'utilisateur U_Z; l'utilisateur U_y introduit peut être une faille dans le système de protection.

Nous proposons donc une règle de propagation des autorisations qui évite ce type de problème en n'augmentant pas les autorisations. La règle définit les variables L_y (O,U) et D_y (O,U), dans l'Espace de Travail Y, comme le minimum de la variable d'autorisation distante D_X (O,U) définie par le serveur X et de la variable Maximun de Y:

$$L_y (O,U_Z) = D_y (O,U_Z) = \min(D_X (O,U_Z) , M_y (O)) \quad \forall U_Z \quad \forall O$$

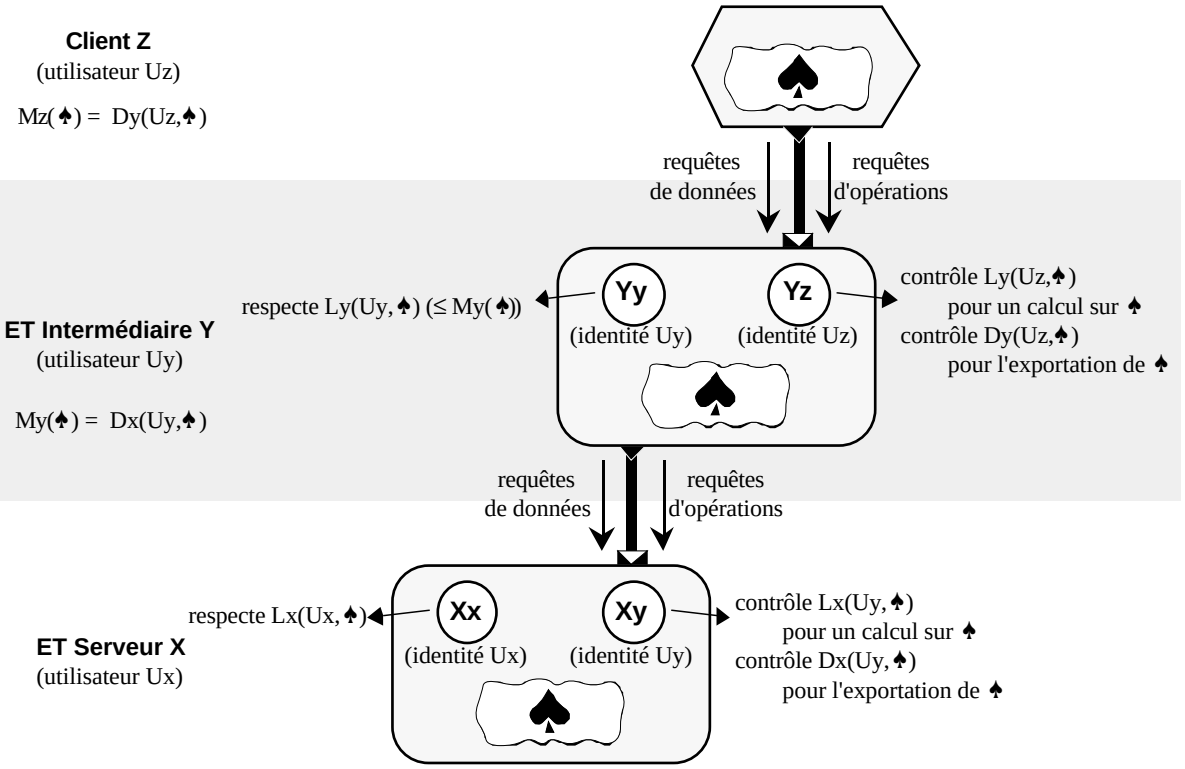
$$(\text{ et se simplifiée à } = D_X (O,U_Z) \text{ si } U_Z=U_y \quad \forall O)$$

Dans la figure 4.10, L'Espace de Travail Y effectue des activités applicatives et des activités de services pour des clients d'identité U_Z :

- l'activité applicative Y_y respecte la variable L_y (O,U_y) lors des accès aux objets O pendant le calcul des transactions,
- l'activité de service Y_Z contrôle les variables L_y (O,U_Z) lors du calcul d'opération et D_y (O,U_Z) lors des exportations de données vers le client Y.

Récursivement, le client Z définit sa variable Maximale M_Z (O) (= D_y (O,U_Z)); il peut à son tour définir les autorisations d'accès par les variables L_Z (O,U) et D_Z (O,U) et les propager à travers un service.

Figure 4.10 : Propagation des autorisations d'accès dans les abonnements.



IX. Bases de Données Coopératives

IX.1. Le Travail Coopératif

Le travail Coopératif (Computer-Supported Cooperative Work) sucite un intérêt grandissant. Le travail coopératif reproduit les mécanismes de travail en commun et de décision prises dans les organisations sociales [Palmer94, Grudin94]. Ce type d'activité peut être facilité par l'existence de réseaux de stations de travail. Dans ce cas, les membres du groupe peuvent communiquer entre des stations de travail distantes, tout en utilisant en commun des outils de coopération informatique. Un exemple courant est constitué par l'éditeur de groupe dans lequel plusieurs participants modifient en commun un document. Dans de nombreux cas, le travail de groupe porte sur un grand nombre de données archivées (par exemple, dans un atelier de CAO ou de génie logiciel). Le travail de groupe doit alors être supporté par le système de bases de données sous-jacent [Kirsche94, Kamita94, Liang94]

Dans un premier temps, nous modéliserons le travail de groupe dans le cadre d'un seul Espace de Travail, c'est à dire sans se soucier de la répartition des coopérants :

- Un participant du groupe coopératif propose des mises à jour de la base de données au superviseur du groupe sous la forme d'une version alternative.
- Le superviseur avise les autres membres du groupe d'une nouvelle proposition. Dans un premier temps, cette proposition n'est pas communiquée à l'extérieur du groupe.
- Le superviseur conclut le travail coopératif du groupe en décidant d'établir une version définitive de la base à partir des différentes versions alternatives. La procédure de décision qu'exécute le superviseur est propre à chaque application et résulte en général d'un consensus entre les participants du groupe.

Ce modèle de travail coopératif met en oeuvre des mécanismes connus de systèmes de bases de données récents. Les versions d'objets ou de tuples sont proposées dans plusieurs systèmes de bases de données relationnels ou orientés objets [Cellary90, Talens93]. Des mécanismes de transactions imbriquées ont été proposés [Moss85].

La première section présente notre modèle de travail coopératif qui est fondé sur la production des versions alternatives par des transactions coopératives. La section suivante examine les différents rôles de l'activité superviseur. Nous verrons ensuite que le travail coopératif peut être découpé en étapes successives. Notre modèle de travail coopératif peut être facilement rendu persistant et, en particulier, résistant aux pannes. Enfin, le service Coopératif distribue le travail du groupe coopératif en permettant à des Espaces de Travail clients d'exécuter des activités coopératives pour produire et consulter des versions alternatives et de dialoguer avec le superviseur.

IX.2. Coopération dans un Espace de Travail

Les mécanismes que nous proposons de définir dans la suite, n'interprètent pas la sémantique des modifications proposées par des participants du groupe coopératif : l'analyse des modifications est laissée au concepteur de l'application coopérative qui définit les procédures associées aux participants et au superviseur. Nos mécanismes du travail définissent les opérations génériques de génération et de modification des versions alternatives de la base ainsi que des mécanismes de communication entre les participants et le superviseur.

Notre approche du travail coopératif suit un modèle centralisé dans lequel le **participant** (ou membre) du groupe coopératif ne communique qu'avec le **superviseur**. Le groupe coopératif est modélisé par un Espace de Travail englobant (cf. chapitre 3 section III.2.3) dans lequel chaque membre du groupe est représenté par une **activité coopérative** et le superviseur par une **activité superviseur** unique dans le groupe coopératif. Une messagerie relie chaque activité coopérative avec l'activité superviseur.

Le travail coopératif porte sur la totalité de la base de données accédée par l'Espace de Travail. Une proposition de modification de la base coopérative est présentée sous la forme d'une **version alternative**. Une version contient les modifications d'un ou de plusieurs objets de la base.

L'activité Coopérative instancie une ou plusieurs **transactions coopératives** successives pour consulter des versions alternatives de la base, pour en modifier ou bien pour en proposer de nouvelles. L'activité Superviseur instancie également des transactions coopératives pour consulter ou modifier les versions proposées par les activités coopératives; cette activité instancie également une transaction "normale" pour modifier la base avec la version définitive.

Les transactions coopératives se différencient des autres transactions (transactions d'applications ou de services quelconques que nous avons déjà étudiées) car elles ne modifient jamais directement la base de données. Les modifications qu'elles proposent sont effectuées dans des versions de la base. La section suivante définit la manipulation de la base et des versions de celle-ci par les transactions coopératives. Nous détaillerons ensuite les fonctions de l'activité superviseur.

IX.2.1 Transactions Coopératives et Versions alternatives

La transaction coopérative est utilisée à la fois par les activités coopératives (l'application du participant) et par l'activité superviseur pour :

- consulter les objets de la base de données initiale ou ceux des versions alternatives qui contiennent des objets modifiés.
- modifier des objets d'une version alternative de la base de données ou créer une nouvelle version.

Au démarrage, une transaction coopérative spécifie la version qu'elle souhaite consulter ou modifier. Les modifications d'une version alternative sont autorisées en fonction de son statut. Une version alternative a un des deux statuts suivants :

- **modifiable**,

  Une transaction coopérative peut modifier des objets dans cette version à condition qu'elle soit seule à le faire et qu'aucune autre transaction ne consulte la version en question.

  Une version peut être modifiable soit uniquement par son auteur, soit par n'importe quel participant du groupe coopératif. En général, une version correspond à un membre du groupe et celui-ci ne souhaite pas que sa version courante soit modifiée sans son accord : ceci revient à la notion de propriété intellectuelle. Toutefois si le propriétaire de la version décidé de cesser d'y travailler, il peut transférer la propriété à un autre membre du groupe ou en faire une version stable.

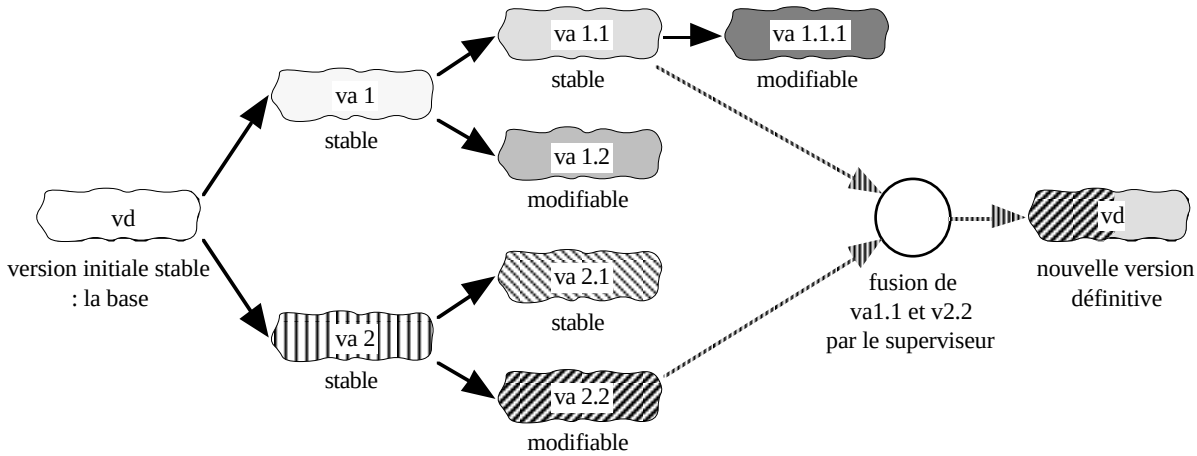
- **stable**

  La version a été marquée stable et elle ne peut plus subir de modifications : une transaction coopérative ne peut que dériver cette version pour produire une nouvelle version modifiable. La version dérivée hérite des modifications déjà validées dans la version stable et peut subir les modifications de la transaction coopérative. La stabilisation est demandée explicitement par l'activité superviseur ou par une des activités coopératives.

  L'intérêt d'une version stable est de permettre la dérivation d'autres versions par d'autres membres du groupe sans porter atteinte à la version de l'auteur initial.

Initialement, la première version est la base qui est considérée comme stable et dérivable par tous. Les différentes dérivations de cette première version et de ses versions dérivées définissent un arbre de versions alternatives dont la racine est la base. Les noeuds interne de ces arbres sont tous des versions stabilisées; ces versions ne peuvent qu'être consultées ou dérivées. Les noeuds feuilles sont soit modifiables, soit stabilisés. Nous verrons dans la section

Figure 4.11 : Arbre de dérivation des versions et Etablissement de la version définitive



suivante que l'activité superviseur maintient une image de cet arbre dans le groupe coopératif.

Dans l'arbre de dérivation de la figure 4.11, une transaction coopérative qui souhaite modifier la version stable va1 (resp. va2 et va2.1) doit préalablement dévirer celle-ci en une nouvelle version va1.3 (resp. va2.3 et va2.1.1) pour réaliser les modifications. Si la transaction coopérative souhaite modifier la version modifiable va1.2, les modification sont directement effectuées sur cette version.

La transaction coopérative se termine soit par la validation soit par l'abandon des modifications proposées. La validation stocke les modifications réalisées dans la version alternative ou crée une nouvelle version dérivée. Après la validation de la transaction coopérative, l'activité coopérative correspondante avise l'activité superviseur des modifications qui sont proposées. Nous verrons dans la section suivante que l'activité superviseur vérifie les modifications proposées et décide d'en aviser les autres activités coopératives.

IX.2.2 Activité Superviseur.

Dans le groupe coopératif, l'activité superviseur est chargée de :

- la **modération des versions modifiées ou dérivées** et la notification de leur existence,
- la **constitution de la version définitive** selon une procédure de décision.

Le contenu de ces deux rôles est spécifique à l'application coopérative puisque l'activité superviseur doit analyser la sémantique des modifications contenues dans les versions alternatives proposées. Néanmoins, ils possèdent une trame générique que nous allons exposer.

Modération et Notification d'une nouvelle version alternative.

Après la validation des modifications d'une version alternative ou de sa création par dérivation depuis une version stable, l'activité coopérative en informe l'activité superviseur. L'activité superviseur exerce alors son rôle de **modérateur** dans le groupe coopératif. La modération d'une version consiste à contrôler le contenu sémantique de la version modifiée ou créée avant de notifier ce changement dans l'arbre des versions auprès des autres activités coopératives.

La notification consiste à adresser, aux autres membres du groupe, le nom et l'auteur de la version modifiée (ainsi qu'éventuellement la date de modification) ainsi que les références de tous les objets modifiés dans cette mise à jour (et éventuellement la localisation de la modification à l'intérieur de ces objets modifiés). La précision de la notification dépend notamment du grain des objets composant la version (voir paragraphe "Grain de Coopération").

L'activité superviseur consulte donc la version modifiée ou créée (au moyen d'une transaction coopérative) et décide alors de :

- Accepter cette version alternative et Notifier le changement.

  L'activité superviseur consulte et approuve le contenu de la version alternative. La transaction superviseur notifie les activités coopératives d'un changement dans l'arbre de versions (i.e ajout d'une version dérivée ou changements subis par une version modifiable).

- Corriger la version alternative et Notifier le changement.

  L'activité superviseur consulte et récuse le contenu de la version alternative. Elle entreprend alors de corriger la version incriminée (au moyen de la transaction coopérative qui consulte la version) puis valide ses corrections. La version corrigée est alors notifiée aux transactions coopératives.

  NB: Il faut mentionner que la correction entraîne un problème de propriété intellectuelle sur la version corrigée : la version est diffusée au groupe avec l'identité du membre qui l'a produite, mais l'auteur ne retrouve pas la version qu'il a proposée au superviseur. La décision de corriger les versions par le superviseur reste possible mais n'est toujours pas acceptable dans un travail coopératif. Une solution intermédiaire est un dialogue entre le superviseur et l'auteur pour demande l'accord de celui-ci sur la modification faite par le superviseur. La version n'est alors diffusée qu'après accord de l'auteur.

- Récuser et Demander la correction de la version alternative.

  Comme dans le cas précédent, l'activité superviseur n'accepte pas la version modifiée ou créée. Elle demande, à l'activité coopérative, de corriger elle même la version avant de lui soumettre de nouveau la version avec les corrections apportées. La demande de correction peut être accompagnée des points du litige qui ont provoqué le refus. Chaque point du litige correspond au couple <objet litigieux, indication de cause du litige>. Dans l'exemple de Journal édité par plusieurs écrivains (i.e. les participants) , le rédacteur en chef (i.e. le superviseur) peut décider qu'un Article (i.e. l'objet litigieux) est trop lourd ou peu clair (i.e. les causes du refus). Quand une version modifiable est refusée, cette version ne peut plus être consultée, modifiée ou dérivée par des activités autre que l'activité coopérative qui doit la corriger.

Contrôle des Accès aux Versions Alternatives.

L'activité superviseur doit donc contrôler les accès des transactions coopératives aux versions alternatives. Avant de consulter ou de modifier une version, une transaction coopérative demande à l'activité superviseur l'autorisation :

- de consulter une version,

  l'activité superviseur vérifie si la version modifiable n'est pas en cours de modification par une autre transaction.

- de modifier une version modifiable,

  l'activité superviseur vérifie si la version n'est pas en cours de consultation par une autre transaction et si la version est modifiable. Une forme de concurrence moins stricte peut être cependant envisagée (par exemple, du type 1 écrivain et N lecteurs).

- de stabiliser une version modifiable.

  l'activité superviseur vérifie si la version modifiable n'est pas en cours de modification par une autre transaction avant de stabiliser la version au profit de l'activité coopérative.

- de modifier ou consulter une version refusée.

  l'activité superviseur vérifie si la transaction coopérative appartient bien à l'activité qui doit corriger cette version.

Ce contrôle des versions correspond à un contrôle de concurrence d'accès aux versions dans lequel le grain de concurrence est la version [Ellis89]. Ce contrôle est centralisé par l'activité superviseur.

Le contrôle est complété par la vérification de la propriété intellectuelle : l'Espace de Travail vérifie avant d'autoriser une dérivation ou une modification si le participant est l'auteur de la version ou si l'auteur a cédés ses droits sur la version.

Grain de Coopération.

L'activité superviseur conclut le travail du groupe coopératif en élaborant une version définitive de la base à partir de la base originale et des versions alternatives proposées par les différentes activités coopératives. La procédure d'élaboration de cette version est spécifique à l'application coopérative mais elle suit en général une des deux lignes de conduites décrites ci-dessous:

(1) Version exclusive

La version définitive de la base est une des versions alternatives proposées. Les autres versions ne sont pas prises en compte lors de l'élaboration. Cette méthode d'élaboration utilise l'intégralité des modifications proposées par la version choisie.

Le choix exclusif d'une version ne correspond pas véritablement à un mécanisme du travail coopératif : Il s'agit plutôt d'un mécanisme de **concours** entre les participants du groupe. Sociologiquement, les participants sont des concurrents qui généralement ne souhaitent pas collaborer ensemble. L'utilisation de mécanismes coopératifs, dans ce cas, est donc discutable.

(2) Compilation de Versions.

La version définitive de la base est une compilation des différentes versions alternatives : la transaction superviseur choisit des objets simples ou composites modifiés dans les différentes versions alternatives pour élaborer la version définitive. Les objets choisis définissent le grain de coopération de l'application.

Le **grain de coopération** est une notion propre à l'application coopérative : il n'a pas de réalité pour les mécanismes de coopération que nous proposons pour le travail coopératif. Le grain de coopération désigne le type des objets de l'application coopérative qui serviront à élaborer la version définitive. L'élaboration de la version consiste à choisir une version de chaque objet de ce type parmi les versions alternatives pour modifier la base. Ces objets peuvent être des objets simples (c'est à dire des unités de manipulation des données dans l'Espace de Travail) ou bien des objets composites (que l'application constitue en structurant des objets simples).L'élaboration est physiquement réalisée par l'activité superviseur.

Illustrons la notion de grain de coopération par l'exemple d'une édition coopérative. L'édition coopérative porte sur un Document, la base de données, qui contient une liste d'objets composites Section. Chaque objet Section est composé d'une liste d'objets

composites Paragraphe. L'objet Paragraphe est une suite d'objets simples Lignes. Les écrivains (i.e. les participants) peuvent modifier n'importe quel objet Ligne dans le document mais le rédacteur en chef (i.e. le superviseur) élabore la version définitive en établissant le grain de coopération dans l'édition.

- S'il décide que le grain de coopération est le Paragraphe, alors il établit la version définitive en utilisant, pour chaque objet Paragraphe du Document, la version de l'objet Paragraphe correspondant dans une des versions alternatives (un objet Paragraphe est dite modifié quand au moins un objet Ligne, le composant, a été modifié).
- S'il décide que le grain de coopération est la Section, le rédacteur ne réalise la version définitive qu'avec des objets Section complets pris dans les différentes versions alternatives.

Les participants du groupe n'ont en principe aucune notion du grain de coopération. Néanmoins, dans un fonctionnement par consensus, le grain de coopération est décidé par le superviseur après avoir fait l'objet d'un dialogue et éventuellement d'une négociation (avec vote) avec les coopérants. Le grain de coopératif peut aussi changer dynamiquement au cours du travail coopératif et être variable dans l'ensemble de la base après négociation. Le dialogue repose sur le mécanisme d'échange de message entre les activités coopérant et l'activité superviseur.

Décision du Choix de la Version Définitive.

Nous venons de voir comment est élaborée la version définitive. Il reste à choisir le grain de coopération dans une version plutôt que dans une autre. Ce choix est laissé à l'activité superviseur qui peut :

- décider seule des versions à choisir,

  L'activité superviseur effectue les modifications en ne suivant que les décisions impératives d'un seul utilisateur dans le groupe, le superviseur. Pour l'édition coopérative d'un journal, cette méthode rend le rédacteur en chef seul maître de la version définitive.

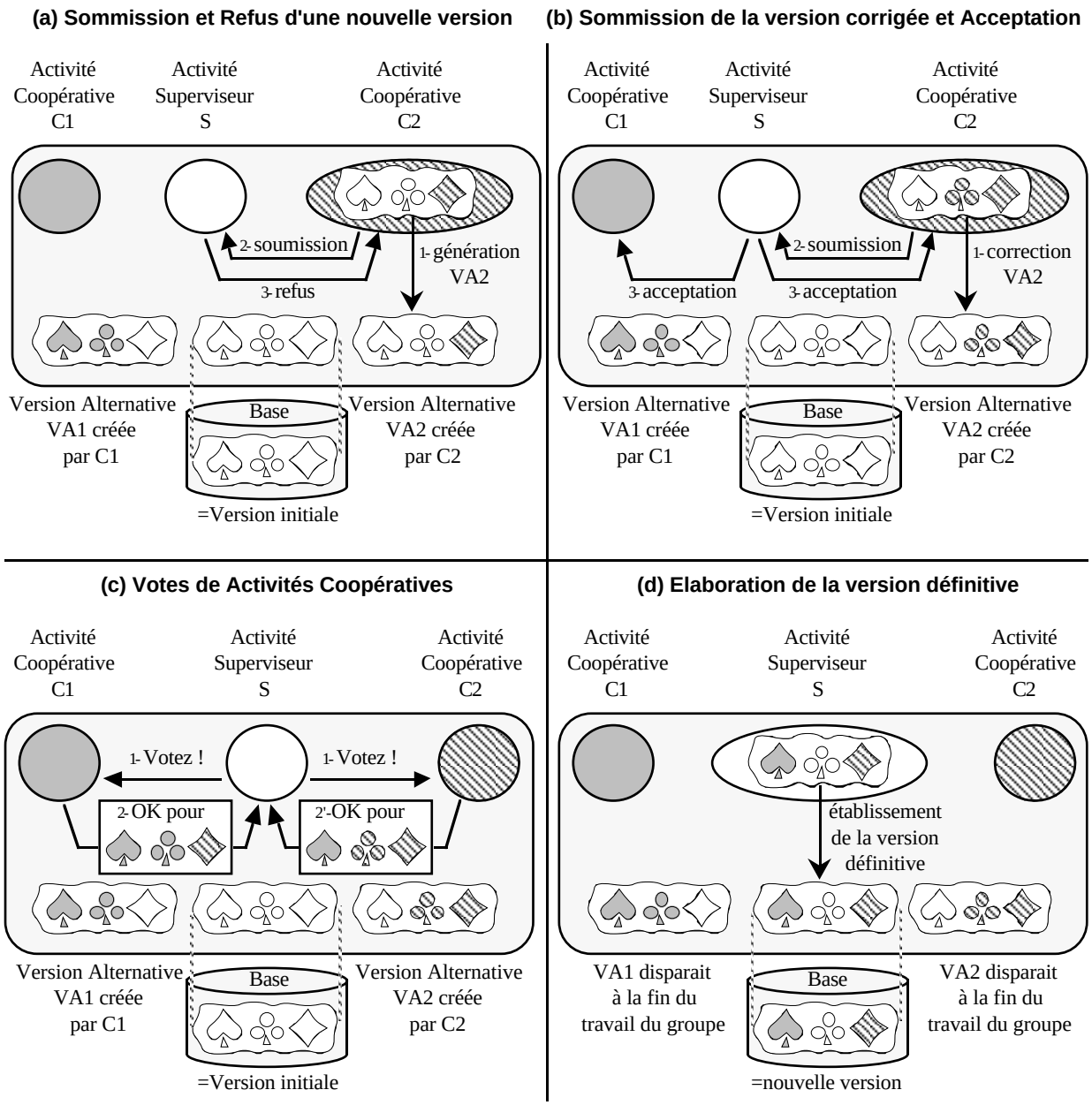
- réaliser un consensus décidé par les participants du groupe.

  L'activité superviseur effectue les modifications en suivant un consensus des participants. Le consensus peut se réaliser à la suite d'un processus d'élection dont les formes peuvent être variées (plusieurs tours, majorités relatives ou absolues …). L'élection est pilotée par l'activité superviseur, qui recueille le vote des participants, et décide de la suite de l'élection en fonction des résultats. L'organisation du vote ne nécessite que des dialogues entre les activités coopératives et l'activité superviseur.

  L'élection peut être précédée par une "campagne électorale" au cours de laquelle un participant peut essayer d'influencer le vote futur des autres en justifiant son propre vote. La campagne électorale engage le superviseur, qui recueille les explications de vote des participants, et les diffuse dans l'ensemble du groupe. Les explications de vote sont des annotations d'une version qui associent un auteur de l'annotation et une version ou un objet d'une version et un commentaire (texte), et/ou un vote (une

note chiffrée). Il peut être également intéressant d'associer des annotations à des niveaux entre la version et le granule de coopération voir à des niveaux inférieurs. Une façon d'implanter les annotations, faites par un auteur sur une version, est d'en faire un objet de même structure que la version. Une autre façon est d'en faire des tuples < pointeur sur l'objet annoté, annotation proprement dite >.

Figure 4.12 : Phases dans le travail de deux activités coopératives.



Exemple.

Illustrons ces différents mécanismes avec l'exemple de la figure 4.12. La base coopérative est initialement composée d'objets blancs. L'activité coopérative C1 a déjà proposé une version alternative VA1. Dans notre figure, les objets Pique, Trèfle et Carreau représentent des objets composites qui servent de grain de coopération à l'application. L'objet composite est modifié si une de ces composantes est modifiée.

(a) L'activité coopérative C2 crée une nouvelle version VA2 et soumet celle-ci à l'activité superviseur S. Cette dernière exerce son rôle de modérateur et refuse de diffuser VA2 au groupe de travail.

(b) L'activité C2 corrige la version VA2 en la modifiant (l'objet Trèfle est modifié) et soumet de nouveau la version corrigée à l'activité superviseur S qui accepte la version et diffuse VA2 au groupe de travail.

(c) L'activité S démarre la consultation des activités coopératives en vue d'élaborer la version finale. Les activités coopératives répondent en votant pour leurs propres modifications complétées des modifications apportées par l'autre.

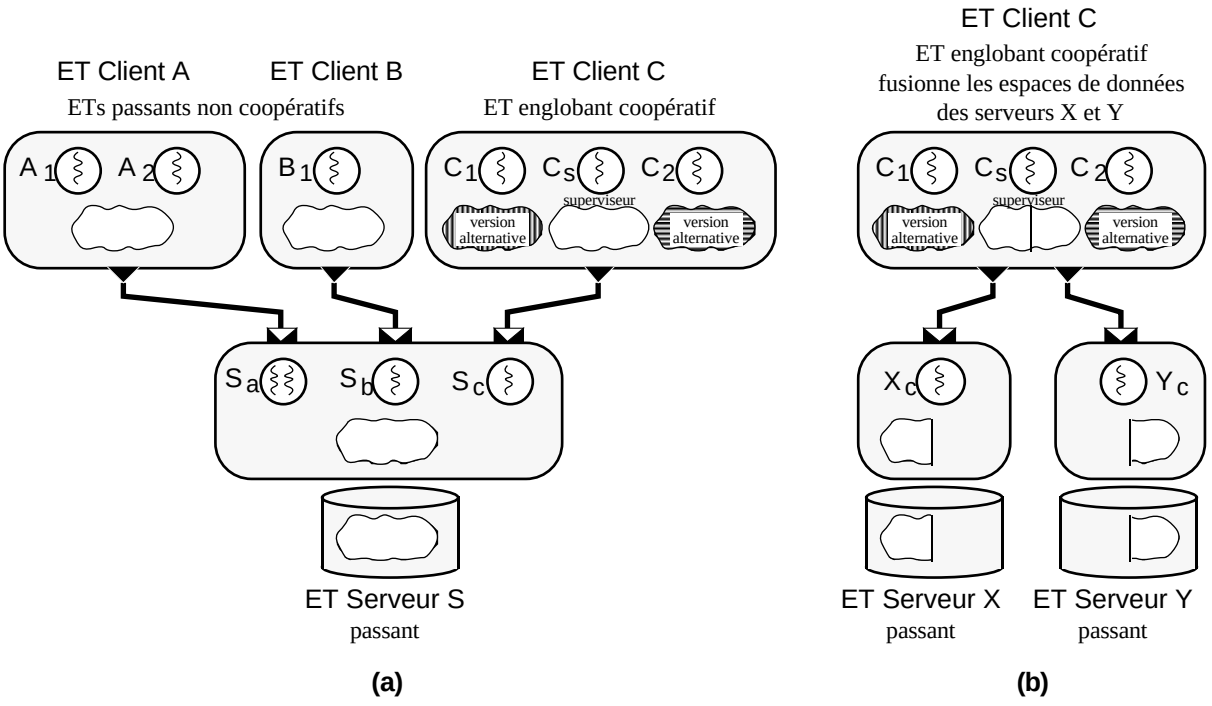
(d) L'activité S, qui a reçu les résultats du vote, décide d'établir la version définitive en fusionnant les modifications qui ont reçu la majorité absolue des suffrages, comme l'objet Pique grisé et l'objet Carreau hachuré. Rappelons que la majorité d'un groupe de deux est deux (la moitié des voix + 1 voix supplémentaire) et que ,dans ces conditions, aucune version de l'objet Trèfle n'a été crédité de la majorité des voix pour être sélectionnée pour la version définitive. L'activité superviseur termine le travail du groupe C1 et C2 en validant cette version dans la base. Les versions alternatives peuvent alors disparaître.

IX.2.3 Travail Coopératif et Etapes Coopératives

Notre modèle de travail coopératif se place dans un contexte où la base de données peut être également accédée par des utilisateurs qui ne font pas partie du groupe coopératif. L'ensemble des consultations et des modifications réalisées à l'intérieur du groupe coopératif doit être perçu de l'extérieur comme une seule transaction.

L'Espace de Travail englobant (cf. chapitre 3 section III.2.4) qui héberge le groupe coopératif se comporte comme une seule transaction englobante encapsulant les consultations et les modifications apportées par les transactions des activités (superviseur ou coopérative). Dans le contexte où la base est accédée à la fois par le groupe coopératif et des utilisateurs externes au groupe, l'Espace de Travail englobant doit se placer en tant que client du serveur non coopératif pour partager la base avec les autres utilisateurs. Dans la figure 4.13.a, l'Espace de Travail englobant C est considéré comme une et seule transaction quelconque par le serveur S de la base. Cet Espace de Travail C partage les accès à la base avec les autres transactions A1, A2 et B1 du système. Dans le cadre d'une base répartie sur plusieurs serveurs (figure 4.13.b), l'Espace de Travail coopératif propose le travail coopératif en fusionnant les services de Données des morceaux de la base comme le fait l'Espace de Travail I dans la section VI.2.2 du chapitre 3.

Figure 4.13 : (a) Accès concurrents à la base par le groupe Coopératif et d'autres utilisateurs non coopératifs.
(b) Accès à une base distribuée par le groupe Coopératif

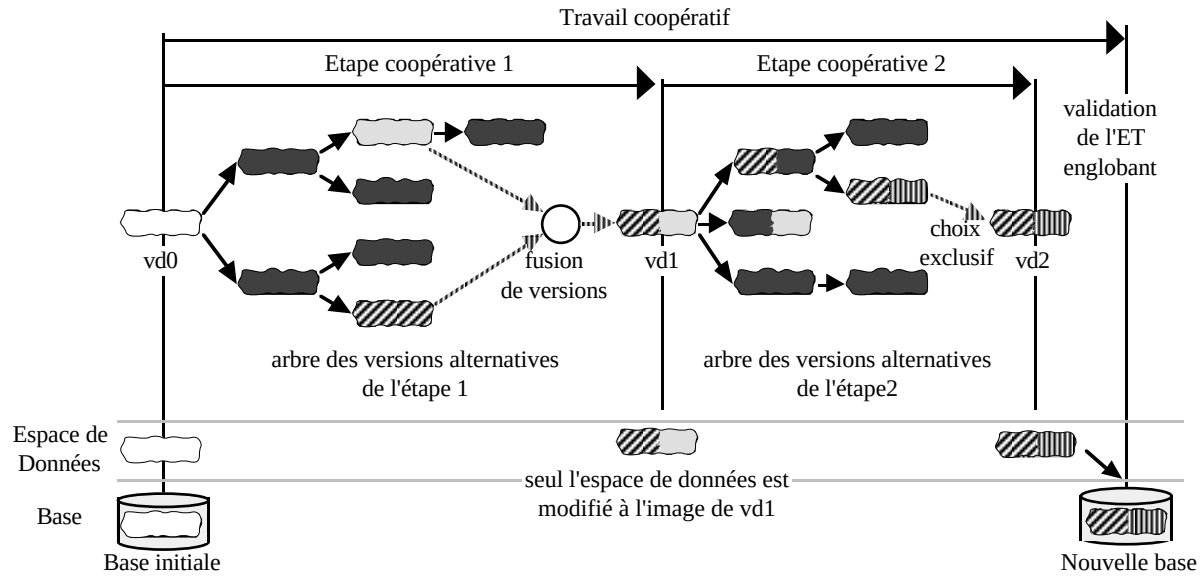


Le travail coopératif est une succession d'**étapes coopératives**. L'étape coopérative correspond à une phase de proposition de versions alternatives par les activités coopératives qui se termine par l'élaboration d'une version définitive par l'activité superviseur. Cette version définitive sert alors de version initiale pour l'étape suivante. L'étape coopérative peut se terminer par l'abandon des versions alternatives produites au cours de celle-ci. Il peut être souhaitable de conserver les versions alternatives précédentes si le consensus n'a été obtenu que sur une partie des objets de la version définitive. Toutefois, cette conservation pose des problèmes d'implantation si les versions alternatives sont réalisées sous forme des versions différentielles par rapport à la version définitive issue du consensus précédent.

La version définitive produite par une étape peut être visible à l'extérieur du groupe coopératif (fonctionnement quasi-passant) ou seulement à l'intérieur du groupe coopératif. La première solution peut être intéressante si la durée de la session du groupe est très longue. L'Espace de Travail englobant termine le travail du groupe en validant la version définitive élaborée lors de la dernière étape dans la base : les modifications apportées sont alors visibles à l'extérieur du groupe coopératif.

La figure 4.14 illustre la succession d'étapes coopératives. Le travail de groupe démarre sur une image originale de la base (en blanc) qui correspond à la version vd0. La fin de la première étape produit une version définitive vd1 en fusionnant plusieurs versions de vd0. La deuxième étape démarre avec vd1 comme version initiale et en abandonnant les versions alternatives de vd0. La deuxième étape se conclut par l'élaboration de la version définitive vd2 à partir d'une seule version alternative. Enfin le travail coopératif se termine sur la validation de l'Espace de Travail englobant qui valide les modifications, apportées par vd2, auprès de la base .

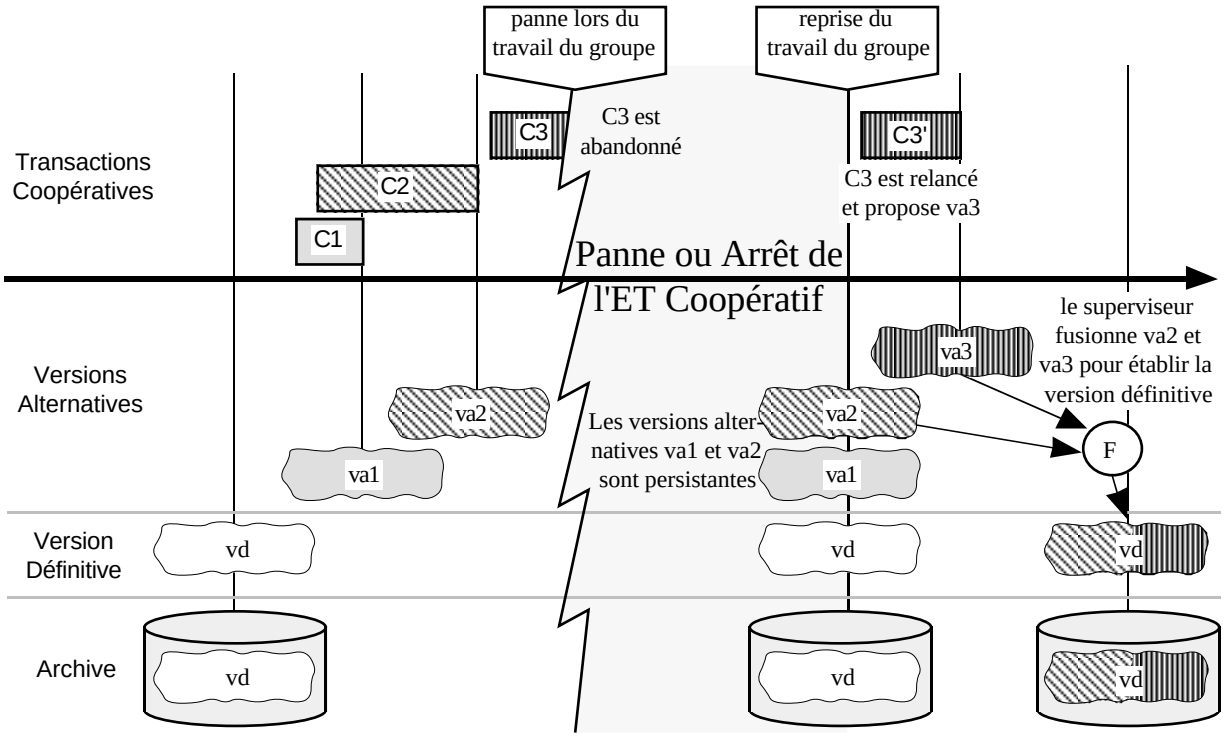
Figure 4.14 : Travail Coopératif et Etapes Coopératives.



IX.2.4 Persistance du travail de groupe

Le travail coopératif peut s'échelonner sur plusieurs heures ou plusieurs jours. Dans un tel contexte, les versions alternatives proposées par les participants ne doivent pas disparaître en cas de panne ou d'arrêt convenu du système. La reprise du système doit pouvoir reconstituer le groupe coopératif et le redémarrer là où il avait été arrêté. Nous proposons de rendre notre

Figure 4.15 : Persistance du travail d'un groupe coopératif.



modèle de travail persistant aux arrêts du système en assurant la persistance pour les versions alternatives proposées par les transactions coopératives. L'activité superviseur doit rendre persistant les informations concernant l'arbre de dérivation. Une fois proposée, une version alternative devient résistante aux pannes et reste visible des seules activités du groupe de travail tant que la version définitive n'a pas été élaborée. Cette fonctionnalité du travail persistant se rapproche des travaux effectués dans Versant sur les transactions persistantes [Chou92].

Dans l'exemple de la figure 4.15, les versions alternatives va1 et va2 sont proposées par les transactions coopératives C1 et C2 avant la panne alors que la transaction C3 est interrompue par la panne avant de proposer va3. La reprise du travail reconstitue le groupe coopératif avec les versions va1 et va2 proposées avant la panne. La transaction C3 est relancée par son utilisateur pour proposer sa version alternative va3. Le travail de groupe se termine normalement par la fusion de versions proposées avant (va2) et après (va3) la panne.

IX.3. Service Coopératif

Dans les sections précédentes, l'ensemble du travail coopératif est réalisé au sein d'un seul Espace de Travail : les participants du groupe coopératif se partagent cet Espace de Travail englobant. Or le travail coopératif doit être envisagé dans des environnements distribués. Le but du service Coopératif est de permettre l'accès d'un coopérateur distant au groupe ou de permettre la coopération de plusieurs sous-groupes.

Le service Coopératif se compose de:

- un service de données.

  Le client peut importer les objets de la base coopérative et les images des objets modifiés dans les versions alternatives. Cette partie du service Coopératif est en fait strictement identique au service de Données : ceci est rendu possible car, comme nous le verrons dans le chapitre 5 section VII, les versions alternatives sont implantées sous la forme d'objets composites quelconques. Chaque objet de version regroupe les objets modifiés par rapport à la version alternative précédente ou par rapport à la base suivant la méthode des "deltas" [Tichy82, Palisser90].

- un service de communication entre les transactions coopératives distribuées et la transaction superviseur centralisée.

  Cette partie du service coopératif permet de réaliser les dialogues entre les activités coopératives distribuées sur les différents clients et l'activité superviseur centralisée sur l'Espace de Travail serveur. Ces dialogues sont nécessaires lors des soumissions de versions et lors du consensus sur la version définitive.

L'originalité du service Coopératif tient donc dans la partie communication que nous allons détailler. Comme dans les autres services, le serveur coopératif abonne des Espaces de Travail clients au moyen d'activités de service. Ces activités de service coopératif se comportent comme des activités coopératives lors des accès à la base et aux versions et en matière de dialogue avec l'activité superviseur : le client est ainsi perçu par le serveur, et par son activité superviseur, comme une et une seule activité coopérative.

Le client coopératif est un Espace de Travail passant (cf. chapitre 3 section III.2.3) organisé suivant le schéma du serveur avec plusieurs activités coopératives dialoguant avec une activité superviseur locale :

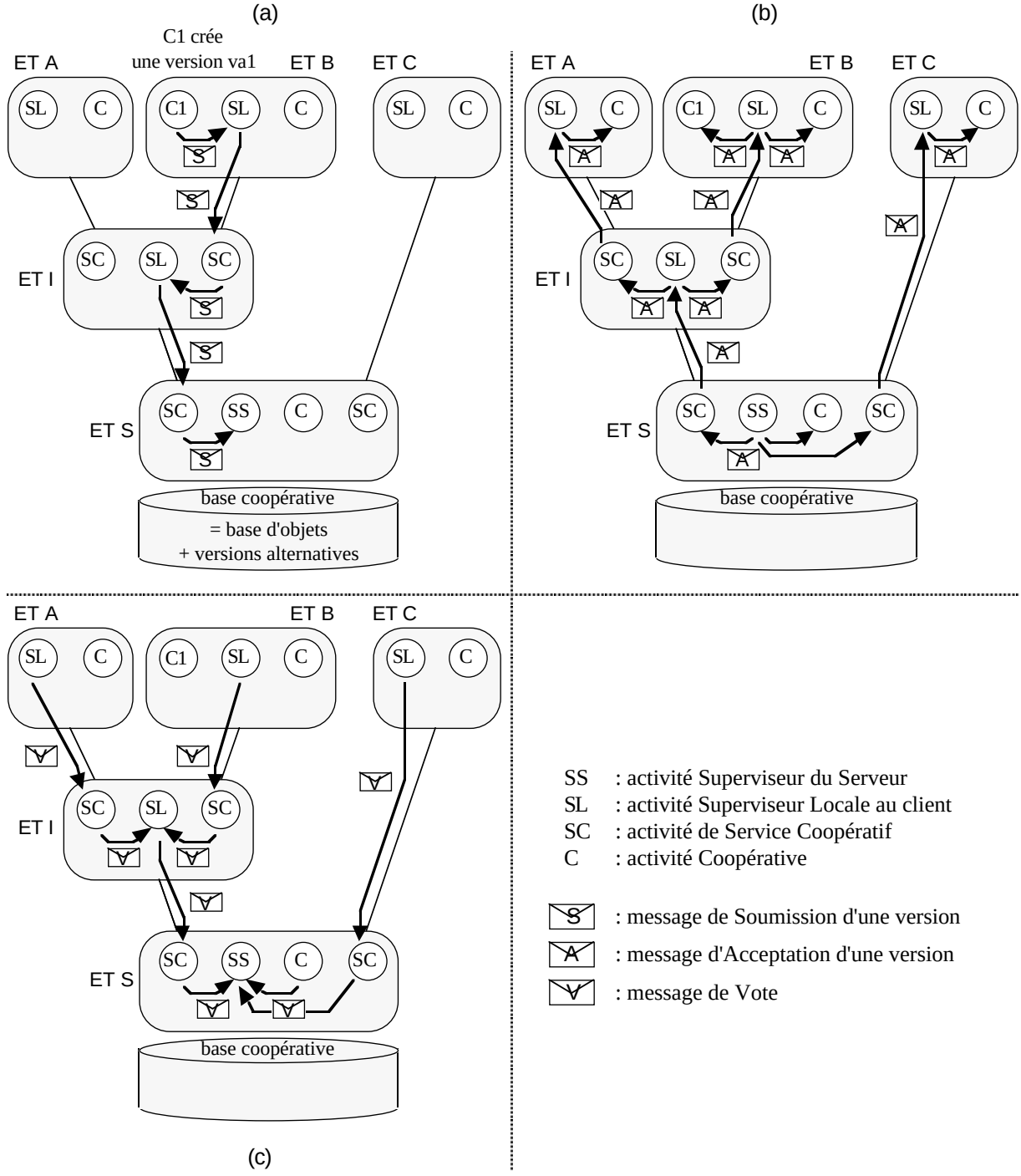
- les activités coopératives consultent, modifient ou créent des versions alternatives de la base au moyen de transaction coopératives. Elles soumettent les modifications proposées à l'activité superviseur locale de l'Espace de Travail client. Le client doit être obligatoirement un Espace de Travail passant pour que les changements, apportés sur les versions, soient répercutés dans l'espace de données de l'Espace de Travail serveur englobant quand une transaction coopérative du client valide (cf. chapitre 3 section IV.2.4).
- l'activité superviseur locale au client relaie le dialogue entre l'activité superviseur du serveur et les activités coopératives du client.

  Dans le cas de la soumission d'une version, l'activité superviseur du client transmet la soumission à l'activité de service; celle-ci la transmet à l'activité superviseur du serveur qui accepte ou refuse la version. En cas d'acceptation, l'activité superviseur serveur notifie les activités de l'Espace de Travail du changement intervenu dans l'arbre des versions. Chaque activité de service remonte cette information vers l'activité superviseur de leur client qui notifie à son tour les activités coopératives client des changements intervenus. En cas de refus, l'activité superviseur serveur informe l'activité de service du refus de la version qui a été modifiée ou créée par son client. L'activité de service remonte le refus vers l'activité superviseur qui en informe l'activité coopérative modificatrice.

Le but de cette activité superviseur client est d'unifier le cas d'un coopérateur distant isolé et celui de la coopération d'un sous groupe de plusieurs membres. Cette activité n'a normalement que le rôle générique de relai dans les dialogues entre les activités coopératives client et l'activité superviseur serveur. Cependant, cette activité peut être utilisé par le concepteur de l'application comme organe décisionnel dans le sous-groupe piloté par le client. Les décisions du sous-groupe peuvent être impératives ou consensuelles comme dans le cadre de l'activité superviseur serveur.

Le service Coopératif est un service récursif : un Espace de Travail coopératif client peut à son tour publier ce service et abonner des Espaces de Travail coopératifs clients au moyen d'activités de service. Cet Espace de Travail intermédiaire permet au concepteur d'introduire la notion de sous-groupe coopératif : l'activité superviseur de cet Espace de Travail reprend les rôles de l'activité superviseur serveur en les appliquant aux activités du sous-groupe. Ainsi, cette activité pilote le sous-groupe en modérant les versions proposées au niveau du sous-groupe et en élaborant une version définitive pour le sous-groupe (avec ou sans consensus des participants du sous-groupe); la version définitive du sous-groupe est considérée par l'activité superviseur serveur comme une version alternative quelconque.

Figure 4.16 : Dialogue dans un travail coopératif distribué.
(a) Envoi de la soumission d'une nouvelle version au Superviseur
(b) Diffusion de l'acceptation de la version aux activités coopératives.
(c) Envoi des votes pour élaborer la version finale.



La figure 4.16 illustre une architecture distribuée pour le travail coopératif. Les Espaces de Travail clients sont abonnés à l'Espace de Travail serveur ou à un Espace de Travail intermédiaire relayant ce service.

(a) Une activité coopérative C1 propose une nouvelle version alternative de la base en la soumettant au superviseur. La soumission est ainsi relayée jusqu'à l'activité superviseur du serveur (SS) par les activités superviseur des clients (SL) et par les activités de service (SC).

(b) L'activité superviseur SS décide d'accepter cette version et diffuse l'acceptation aux autres activités du serveur. Les activités de service relayent cette acceptation aux activités SL de leur client qui à leur tour diffusent cette information aux autres activités du client.

(c) L'élaboration de la version définitive est réalisée à la suite d'un vote distribué. L'activité superviseur SS réalise la collecte des votes des activités coopératives. Sur les Espaces de Travail clients, la décision est confiée seulement à l'activité superviseur locale qui traduit le pouvoir de l'utilisateur de l'application instanciée sur le client. L'activité superviseur, locale de l'Espace de Travail intermédiaire, relaie par défaut les décisions des clients A et B, mais elle pourrait être utilisée pour définir un sous-groupe coopératif composé de A et B.

X. Conclusion

Dans ce chapitre, nous avons présenté les utilisations du modèle des Espaces de Travail formalisé dans le chapitre 3. Les utilisations répondent aux critiques formulées à l'encontre des Gérants d'Objets Persistants :

- Leurs architectures manquent de flexibilité pour s'adapter à la structure sous-jacente au système d'information de l'entreprise.
- Ils ne proposent qu'un seul mode d'interaction, la Migration de Données, qui ne peut pas être utilisée sans risque dans les environnements nécessitant de sécuriser les données et les opérations.

Dans les sections II à IV, nous avons répondu à cette première série de critiques en construisant les architectures existantes de Bases de Données à partir du service de Données. Les sections VI à VIII offrent une solution aux problèmes des architectures sécurités. Enfin la section IX termine ce chapitre en proposant d'élargir le champ d'application des Gérants d'Objets au travail coopératif.

La section II présente l'utilisation la plus simple de l'Espace de Travail : il s'agit d'une base accédée localement dans le cadre d'une machine isolée (utilisateur isolé ou machine embarquée); ce contexte correspond aussi aux phases de mise au point des applications par un développeur.

La section III introduit l'usage de services dans l'Espace de Travail : un Espace de Travail propose un accès aux données privées sous la forme du service de Données; un autre Espace de Travail peut s'abonner à ce service pour compléter son espace de données par celui du "serveur". L'introduction du service de Données rend possible l'accès à des données persistantes réparties sur les machines du réseau. Néanmoins, cette approche laisse des applications côtoyer des services dans un même Espace de Travail, ce qui a l'inconvénient de ne pas garantir une continuité du service.

La section IV propose de reproduire l'architecture Client-Serveur qui sépare les applications des "organes" d'archivage des données; les premières sont exécutées par des Espaces de Travail clients qui s'abonnent à un ou plusieurs Espaces de Travail serveurs pour importer les données persistantes. Nous avons modéliser l'approche directe qui laisse le client se connecter à l'ensemble des serveurs qui lui sont nécessaires. L'approche symétrique pair-vers-pair contraint le client à ne dialoguer qu'avec un seul serveur; celui-ci sert ses propres données et se charge d'importer les autres depuis ses pairs, les autres Espaces de Travail serveurs du réseau. Nous avons comparé les deux approches sur le plan des communications et suivant les différentes approches de conception des applications. Cependant, l'approche symétrique permet de réaliser de manière simple un contrôle (centralisé) de contraintes d'intégrité par le serveur.

La section V illustre un des objectifs majeurs du modèle des Espaces de Travail : la possibilité du jouer le rôle de relai dans un service afin d'ajuster le Gérant d'Objets à l'architecture matèrielle. Le système informatique de l'entreprise repose sur un hiérarchie de réseaux (locaux et distants) et de machines (stations de travail, serveurs départementaux, mainframe). L'architecture ajustée du gérant d'objets comporte alors des Espace de Travail intermédiaire dont le but est factoriser les accès aux serveurs distants pour les applications exécutées sur les stations d'un même réseau local.

La section VI présente un autre type de service, le service d'Opérations. Le but de ce service est de faire exécuter des opérations complexes par le serveur. Le mode d'interaction entre le client et le serveur est la Migration des Requêtes (Query-Shipping). Ce type de service garantit la sécurité des traitements et la confidentialité des données car l'application ne manipule pas directement les données. Il évite aussi à un client de pénaliser le système par des traitements locaux peu sélectifs.

La section VII propose un service qui permet aux deux types d'interaction de se côtoyer dans la réalisation d'une application. Le client qui exécute l'application peut importer des données pour les consulter et les modifier mais également demander aux serveurs d'effectuer des opérations sur ces mêmes données. Le service veille principalement à la cohérence des données entre l'espace de données du client et le contexte d'exécution des requêtes sur le serveur.

La section VIII redéfinit les autorisations d'accès sur les données dans un Espace de Travail dans le cadre des services. Les autorisations d'accès aux données sont distinguées quand l'opération (consultation ou modification) est réalisée par le serveur (i.e. dans le cadre d'une requête) ou par le client (i.e. après avoir importé la donnée). Cette distinction permet de sécuriser les traitements tout en laissant les consultations possibles par le client ou bien protéger la confidentialité d'une partie des données tout en permettant au serveur de l'utiliser dans le calcul des requêtes.

Finalement, la section IX utilise l'Espace de Travail pour coordonner et héberger le travail d'un groupe de coopérateurs. Nous avons défini un modèle centralisé pour le travail coopératif. Cependant le travail est distribué sur un réseau de machines : les participants proposent des versions alternatives de la base; le superviseur du groupe diffuse les versions produites aux autres membres du groupe et termine le travail du groupe par l'élaboration de la nouvelle version de la base à partir d'un consensus des participants.

La table 4.17 résume les propriétés des différents services proposés par les Espaces de Travail. La récursion est la possibilité pour un client de publier le même service et de servir de relai. Nous verrons dans le chapitre 5 (Implantation des Espaces de Travail) que le type d'appel du client, la récursion et le nombre de transaction influenceront la structure de l'Espace de Travail. L'usage spécifique du service signifie que la définition d'un service est réalisée en corrélation avec celui de l'application qui l'utilise. Cependant, un service d'Opération peut très bien être à usage général comme dans le cas d'un serveur de langage de requêtes.

Table 4.17 : Propriétés des Abonnements aux différents services.

Service	Trafic	Appel Client	Usage	Récursion	Transaction
Données	Données	asynchrone ré-entrant	général	Oui	plusieurs par client
Coopératif	Données et Notifications	asynchrone ré-entrant	spécifique	Oui	plusieurs par client
Opération	Questions	asynchrone ré-entrant	spécifique	Oui	plusieurs par client
Mixte	Données et Questions	synchrone	spécifique	Non	une seule par client