ANALYSE DES PROTOCOLES DE CONTROLE DE CONCURRENCE ET DES PROPRIETES ACID DANS LES SGBD TEMPS REEL

Claude Duvallet* — Zoubir Mammeri — Bruno Sadeg***

** LIH, Faculté des Sciences et Techniques*
25, rue Philippe Lebon, 76058 Le Havre Cedex
{duvallet, sadeg}@fst.univ-lehavre.fr

*** IRIT, Université Paul Sabatier*
118, route de Narbonne, 31062 Toulouse Cedex
mammeri@irit.fr

Résumé. Il existe des applications temps réel qui manipulent des quantités importantes de données sur lesquelles sont effectués des traitements contraints par le temps. La gestion de ces applications pourrait être facilitée par l'utilisation de SGBD temps réel, qui sont des systèmes dont l'objectif est de gérer efficacement les données tout en respectant les contraintes temporelles individuelles des transactions et des données. Depuis une dizaine d'années, plusieurs travaux concernant certains aspects des SGBD temps réel ont été effectués. Ils sont axés notamment sur le contrôle de concurrence des transactions et leur ordonnancement ainsi que sur le maintien de la cohérence des données. L'objectif de cet article est d'effectuer une analyse de ces protocoles afin de voir dans quelle mesure ils peuvent être appliqués à un contexte temps réel strict critique. Des caractéristiques des transactions temps réel plus souples que les propriétés ACID des transactions dans les SGBD traditionnels sont aussi discutées.

Mots-clefs : SGBD, temps réel, contraintes temporelles, contrôle de concurrence, ordonnancement, propriétés ACID.

1. Introduction

Les applications temps réel sont généralement composées d'un système contrôleur (le système informatique) et d'un système contrôlé (l'environnement de l'application). Elles se distinguent des applications traditionnelles par les contraintes temporelles qu' elles doivent respecter et qui sont matérialisées sous forme d' échéances des actions et/ou de durées de validité des données.

Certaines de ces applications manipulent des quantités importantes de données (qui sont, en général, des paramètres représentant les caractéristiques de l' environnement) l'utilisation d'un SGBD (Système de Gestion de Bases de Données) peut s' avérer nécessaire pour gérer ces données de manière efficace [MAM 85]. Cependant, les SGBD traditionnels ne permettent pas de répondre aux besoins de ces applications [RAM 93] car ils n' intègrent pas de mécanismes qui permettent de prendre en compte les contraintes temporelles ; leur objectif en termes de performances étant de minimiser *le temps de réponse moyen* des transactions.

Depuis une dizaine d' années, des travaux sur la conception d'une nouvelle génération de SGBD ont commencé à voir le jour [ABB 88, SON 88, RAM 93, BES 94, BES 96, GRA 92] : ce sont les SGBD temps réel (SGBDTR), qui devraient utiliser les progrès enregistrés dans la conception des SGBD traditionnels (relationnels et objets notamment) et ceux enregistrés dans la conception des systèmes temps réel. L' exemple suivant illustre un domaine d' utilisation potentiel de SGBD temps réel.

On considère un système de contrôle de trafic aérien [PEN 96] dont la complexité s' accroît avec la taille et le volume du trafic de l' espace aérien contrôlé. Une telle application consiste en un système distribué composé de dizaines de serveurs connectés via un réseau LAN à des centaines de stations de travail pour le contrôle du trafic. Les serveurs sont interfacés avec des systèmes externes (radars, stations météorologiques et autres systèmes de contrôle de trafic aérien). Chaque radar doit pouvoir fournir 100 à 200 rapports à chaque scrutation. Les serveurs remplissent différentes fonctions. Certains exploitent les données des radars et déduisent les trajectoires des avions. D' autres enregistrent les données sur des supports pour des analyses ultérieures. Les stations de travail affichent des informations en temps réel (trajectoires, alertes, météorologie, …) pour les utilisateurs concernés. L' application que l'on vient de décrire manipule des quantités importantes d' informations et la plupart des actions citées doivent fournir des résultats dans les temps. Si certains résultats ne sont pas obtenus à temps, le trafic serait déréglé et conduirait probablement à des catastrophes (collision entre des avions, …).

D' autres applications qui pourraient bénéficier d' un support de type SGBD temps réel sont les systèmes de commutation en téléphonie, les systèmes boursiers, les systèmes de contrôle de procédés industriels, les systèmes de téléguidage d'engins, etc.

Parmi les données traitées par les applications temps réel, certaines possèdent des durées de validité (échéances des données [XIO 96a]) au-delà desquelles elles deviennent obsolètes. Typiquement, ce sont les valeurs des paramètres représentant l' état de l' environnement et acquises à partir de capteurs distribués.

D'autre part, les transactions sont soumises à des contraintes temporelles provenant de deux sources différentes :

- des échéances dictées par l'environnement des applications,
- des échéances déduites de la validité temporelle des données.

Contrairement à la gestion d' une base de données traditionnelle, les données dans une base de données temps réel doivent être gérées de telle façon qu' elles soient non seulement cohérentes de manière logique (c'est-à-dire qu'elles doivent satisfaire aux règles d'intégrité), mais aussi du point de vue temporel [SON 92b].

Les transactions dans un SGBD temps réel sont classées selon les conséquences du manquement de l'échéance des transactions [HAR 92, PUR 94, RAM 93] sur l'application et son environnement :

- Les transactions à échéances strictes critiques : une transaction qui rate son échéance peut avoir des conséquences graves sur le système ou sur l'environnement contrôlé ;
- Les transactions à échéances strictes non critiques : si une transaction rate son échéance, elle devient inutile pour le système. Elle est donc abandonnée ;
- Les transactions à échéances non strictes : si une transaction rate son échéance, le système ne l'abandonne pas immédiatement car elle peut avoir une certaine utilité pendant un certain temps encore après l'expiration de son échéance, mais la qualité de service qu' elle offre est moindre.

D'autre part, les SGBD temps réel manipulent deux types de données : les données temporelles qui sont des données ayant des durées de validité, et les données non temporelles qui sont les données rencontrées aussi dans les bases de données traditionnelles. L' objectif de cet article est de faire une analyse des protocoles de contrôle de concurrence des transactions et de l' application des propriétés ACID (Atomicité, Cohérence, Isolation, Durabilité) de transactions dans les SGBD temps réel.

Dans le paragraphe 2, nous présentons les caractéristiques des SGBD temps réel. Puis nous abordons la notion de cohérence dans une base de données temps réel. Les caractéristiques des données manipulées par un SGBD temps réel et des transactions sont ensuite présentées. Dans le paragraphe 3, nous analysons les principaux protocoles de contrôle de concurrence des transactions temps réel. Dans le paragraphe 4, nous discutons de l' application des propriétés ACID dans un contexte temps réel. Le paragraphe 5 est consacré aux transactions à échéances strictes critiques, notamment aux voies de recherche actuelles préconisées pour pouvoir respecter ces contraintes. Nous concluons sur les suites à donner à ce travail et sur la nécessité de concevoir et réaliser des prototypes pour valider les solutions des différents problèmes posés dans les SGBD temps réel.

2. Données et transactions dans les SGBD temps réel

Dans une application temps réel, les systèmes servent souvent à contrôler des dispositifs physiques ou des environnements. Ils ont donc besoin de stocker et de manipuler des volumes importants d'informations qui proviennent de ces environnements. Une grande partie de ces informations sont des données temps réel (qui possèdent des durées de validité ou des échéances et qui ne sont utiles au système que si

elles sont utilisées dans les temps). Les conséquences de l'accès d'une transaction à une donnée après l' expiration de son échéance dépend des besoins particuliers de l'application et de la sémantique des données. Voici un exemple, tiré de [XIO 96b] illustrant les échéances de données. Soit T une transaction de lecture et deux objets Y et Z (Figure 1). La transaction T a besoin de lire deux objets temporels Y et Z pour produire un résultat. Les lectures (à partir de l'environnement) se font aux instants t_b et t_c respectivement. On dispose des éléments suivants :

- Echéance de T : l'instant t_f (initialement égal à 13),
- Durée de validité de Y : 7 unités de temps,
- Durée de validité de Z : 3 unités de temps,
- La transaction T commence à l'instant $t_a = 1$.

A l'instant $t_b = 4$, la transaction lit Y. Son échéance devient $t_f = t_b + 7 = 11$ (car au-delà de t = 11 la donnée Y n'est plus valide). À l'instant $t_c = 6$, elle lit la donnée Z. Son échéance devient $t_f = t_c + 3 = 9$. Si T ne se termine pas avant t = 9, elle est abandonnée.

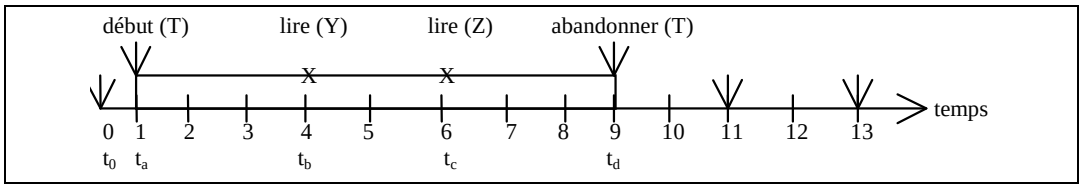

Figure 1. *Exécution d'une transaction temps réel*

L'objectif global d'un SGBD temps réel est double :

- l' exécution des transactions à échéances strictes critiques avant leurs échéances,
- la minimisation du nombre des autres transactions (à échéances strictes non critiques ou à échéances non strictes) qui ratent leurs échéances.

Les principaux travaux actuels sur les SGBD temps réel portent sur les systèmes où les transactions possèdent des échéances strictes non critiques ou des échéances non strictes. Ces travaux concernent particulièrement les protocoles de contrôle de concurrence des transactions et leur ordonnancement dans le but de minimiser le nombre de transactions qui ratent leurs échéances.

Pour les transactions à échéances strictes critiques, il n'existe pas à notre connaissance d'algorithmes d'ordonnancement des transactions qui garantissent leur terminaison avant échéance. La principale raison provient de l'« imprévisibilité » de ces transactions et du manque de méthodes permettant d'estimer avec précision le temps d'exécution des transactions. Il existe cependant quelques tentatives de modélisation des ordonnanceurs de transactions à échéances strictes critiques, sur lesquelles nous reviendrons au paragraphe 5 [SON 93, BES 94, BRA 95a].

Les objets manipulés par un SGBD temps réel sont classés selon deux critères :

- les objets temporels (dont la validité dépend du temps) et les objets non temporels.
- les objets de base et les objets dérivés. Les *objets de base* représentent des entités concrètes de l'environnement (température, pression, …). Leurs valeurs sont mises à jour par les transactions qui lisent ces valeurs depuis des capteurs ou d'autres

périphériques (terminaux, ...). D'autres transactions peuvent dériver de nouvelles données (ou objets dérivés) à partir de ces données de base. Par exemple, dans une usine automatisée, les données acquises par les robots (position et vitesse des objets en mouvement) sont utilisées pour dériver sa nouvelle trajectoire.

3. Analyse des protocoles de contrôle de concurrence et d'ordonnancement des transactions temps réel

3.1. *Problématique*

Dans les SGBD traditionnels, l'ordonnancement sérialisable – une exécution concurrente de transactions validées est sérialisable si et seulement si il existe une exécution en série équivalente – est le critère accepté par tous pour le maintien de la cohérence de la base en cas d'accès concurrents. Dans les SGBD temps réel, tous les ordonnancements sérialisables ne sont pas acceptables : ceux qui ne respectent pas les contraintes temporelles des transactions sont rejetés. Par conséquent, les techniques de contrôle de concurrence des transactions développées pour les SGBD traditionnels ne sont pas directement applicables dans les SGBD temps réel. Par ailleurs, il est nécessaire de maintenir la cohérence temporelle des données en plus de leur cohérence logique. D'autre part, les protocoles conçus pour l'ordonnancement des tâches temps réel ne peuvent pas être appliqués directement aux transactions des SGBD pour intégrer les contraintes temporelles car les tâches sont différentes des transactions [GRA 92].

On distingue généralement les techniques optimistes des techniques pessimistes.

3.2. *Techniques optimistes*

Dans les systèmes où ces techniques sont implantées, toutes les transactions sur la base sont servies dès qu'elles en font la demande. Mais les écritures sont différées jusqu'à l'instant de la validation de chaque transaction. De plus, avant sa validation, chaque transaction doit passer un test de certification afin de détecter si elle est en conflit avec d'autres transactions qui auraient effectué la validation depuis le début de son exécution.

3.2.1. *Le protocole OCC-BC (Optimistic Concurrency Control - Broadcast Commit)*

Dans [HAR 92], Haritsa et al. utilisent une variante de la validation « en avant », OCC-BC, qui tient compte des priorités des transactions. Considérons une transaction T_c en phase de validation. Si, avant sa validation, cette transaction entre en conflit avec d'autres transactions T_1, T_2, …, T_n et si priorité (T_c) > priorité $(T_i)_{i=1,...,n}$ alors les transactions T_i sont abandonnées, sinon s'il existe au moins une transaction T_k telle que priorité (T_k) > priorité (T_c) alors T_c doit attendre jusqu'à la validation des transactions T_i avec lesquelles elle est en conflit.

L'avantage est que pendant son attente, la transaction T_c garde les ressources qu'elle avait acquises. Un inconvénient potentiel est que les transactions abandonnées sont redémarrées plus tard, et ont donc moins de chances de se terminer avant leurs échéances.

3.2.2. *Le protocole wait-50*

Ce protocole [HAR 92] est conçu pour pallier les problèmes rencontrés dans le protocole OCC-BC. En reprenant la notation du paragraphe précédent, la transaction T_c doit se mettre en attente jusqu'à ce que moins de 50% des transactions $T_{i\ (i=1,...,n)}$ aient des priorités supérieures à la sienne. Une fois que ce stade est atteint, les transactions T_i restantes sont abandonnées sans tenir compte de leurs priorités, et la transaction T_c réalise sa validation.

3.2.3. *Le protocole CCA (Cost Conscious Approach)*

Dans [CHA 98], Chakravarthy et al. proposent l'algorithme CCA d'ordonnancement temps réel basé sur deux algorithmes d'ordonnancement statiques : EDF-HP (Earliest Deadline First -High Priority) et EDF-CR (EDF - Conditional Restart) [ABB 88]. La méthode CCA est basée sur une approche dynamique. Elle comporte une pré-analyse des transactions permettant de connaître la structure de celles-ci et les données accédées selon les chemins d'exécution. Cette étape aboutit à la construction d'un arbre de décision dont les noeuds sont utilisés pour détecter les conflits éventuels en cours d'exécution.

L'estimation des coûts permet d'assigner les priorités aux transactions de façon plus équitable que d'autres algorithmes. Elle s'effectue au moyen d'une heuristique comparable à celle de l'algorithme A^* utilisé en intelligence artificielle et basée sur l'utilisation des points de décisions de la transaction. Le contrôle des accès concurrents se fait de façon dynamique afin de n'abandonner que les transactions qui, au cours de l'exécution, rentrent en conflit. Cette technique permet de réduire le nombre de transactions abandonnées de façon considérable. Les résultats des simulations effectuées par [CHA 98] montrent l'efficacité de celle-ci par rapport aux algorithmes EDF-HP et EDF-CR, notamment dans le domaine des bases de données en mémoire secondaire. Le protocole CCA s'applique essentiellement aux transactions temps réel non strictes et strictes non critiques. Les principaux points positifs de CCA sont l'aspect dynamique, la réduction des redémarrages excessifs rencontrés dans d'autres algorithmes comme EDF-HP [ABB 88] et une distribution plus équitable des priorités. Le point négatif est qu'il ne tient pas compte du coût de la pré-analyse des transactions malgré les nombreuses estimations de coûts effectuées.

3.3. *Techniques pessimistes*

Dans les SGBD traditionnels, l'algorithme de verrouillage à deux phases, 2PL (Two Phase Locking) [ESW 76] est le plus répandu. Son principe repose sur le fait qu'une transaction qui a besoin d'un objet devra d'abord acquérir un verrou sur cet objet. A

chaque fois qu'une transaction T_d demande un verrou sur un objet détenu par une transaction T_a dans un mode conflictuel (lecture/écriture ou écriture/écriture), alors T_d est bloquée jusqu'à la libération du verrou par la transaction T_a. L'adaptation de cet algorithme au contexte temps réel a donné naissance à deux principales approches : *abandon par priorité* et *héritage de priorité.*

3.3.1. *L'abandon par priorité*

L'algorithme [HAR 92] consiste à abandonner la transaction de plus basse priorité quand le phénomène d'*inversion de priorité* survient. Ceci assure que les transactions de plus haute priorité ne soient pas retardées par des transactions de plus basses priorités et assure donc que les transactions urgentes se terminent dans les temps. L'inconvénient est que des transactions peuvent être abandonnées par une transaction de plus haute priorité qui, plus tard, ne peut pas se terminer avant son échéance. Ce qui conduit à une perte de travail et donc à une dégradation des performances du système.

3.3.2. *L'héritage de priorités*

Cet algorithme [HAR 92] permet d'assigner une nouvelle priorité à toute transaction accédant à un objet partagé, lui permettant de s'exécuter à un niveau de priorité plus élevé que ceux des transactions qu'elle bloque. On parle alors d'héritage de priorité. Seuls les conflits réels sont considérés. Avec ce protocole, à chaque fois que le phénomène d'inversion de priorité survient, la transaction détentrice du verrou voit sa priorité augmentée et ramenée au niveau de celle de la transaction demandeuse du verrou, jusqu'à ce qu'elle se termine et libère le verrou. Les transactions détentrices de verrous pourraient donc se terminer plus tôt, réduisant ainsi les temps d'attente des transactions de hautes priorités. Les inconvénients de ce protocole résident dans le temps de blocage des transactions de haute priorité qui reste imprévisible même s'il est réduit, les blocages chaînés qui se produisent quand une transaction à haute priorité est bloquée plusieurs fois durant son exécution par des transactions de plus basses priorités, et les transactions de faible priorité ayant hérité de priorités plus hautes risquent de concurrencer les transactions à haute priorité sur l'accès aux autres ressources du système (CPU, E/S, …) et qui peuvent les conduire à rater leurs échéances.

Une variante de ce protocole [HUA 91] est basée sur l' héritage conditionnel de priorité. Une transaction de faible priorité n'hérite d'une haute priorité que dans le cas où elle est proche de sa terminaison. L'avantage est qu'il y a moins de travail perdu quand la transaction est loin de sa terminaison, et les délais d'attente sont réduits pour les transactions de haute priorité. Ce protocole présente de meilleures performances que celui de l'héritage de priorité simple ou que celui de l'abandon par priorité. Par contre, il nécessite de définir la notion de proximité pour la terminaison d'une transaction.

3.3.3. *Le protocole de plafonnement de priorités (priority ceiling protocol)*

Dans ce protocole, une transaction qui accède à un granule se voit attribuer la plus haute priorité correspondant aux priorités des transactions susceptibles d'accéder à ce

granule. Quand cette transaction libère le granule, elle reprend la priorité qu'elle avait juste avant d'accéder au granule. Il s'agit d'un protocole qui supprime les interblocages et limite le temps de blocage des transactions à priorité élevée à une durée moindre que celle de la durée d'exécution d'une transaction [BES 96]. C'est l'un des seuls protocoles de contrôle de concurrence à ne pas abandonner les transactions. Ses inconvénients sont les suivants :

- il ne donne qu'une borne statique sur la durée de toute inversion de priorité. Cette borne est indépendante du comportement à l'exécution de la transaction,
- la promotion d'une transaction de faible priorité vers une priorité plus haute est déterminée de manière statique (en se basant sur tous les accès potentiels des transactions aux données).

3.4. *Autres protocoles*

3.4.1. *Protocole hybride des intervalles d'estampilles*

Ce protocole [SON 92a] combine les techniques optimistes et l'intervalle d'estampilles. Il utilise la validation « en avant » qui résout partiellement le problème d'utilisation inefficace des ressources en employant la détection « au plus tôt » (le test de certification est effectué durant la phase de lecture [HAR 90]). De plus, ce protocole utilise l'allocation dynamique d'estampilles [BAY 82], ainsi que l'ajustement dynamique de l'ordre de sérialisation en utilisant les intervalles d'estampilles [BOK 87]. Ceci améliore la détection et la résolution « au plus tôt » des exécutions non sérialisables et réduit les abandons inutiles. Le principe de ce protocole est de séparer l'ensemble des transactions en conflits de l'ensemble des transactions qui ne le sont pas. Dans l'ensemble des conflits, on distingue les conflits compatibles des conflits non compatibles [SON 92a]. L'ordre de sérialisation est construit dynamiquement à chaque fois que surviennent des conflits. Pour ajuster cet ordre d'estampilles, un intervalle d'estampilles est assigné à chaque transaction sur la base de son échéance. Il est ajusté chaque fois que la transaction lit ou écrit un objet, pour préserver l'ordre de sérialisation induit par les transactions ayant effectué leur validation. Les transactions sont exécutées selon leur ordre d'estampille.

3.4.2. *Le protocole Contrôle de Concurrence Multiversions (MCC)*

Dans ce protocole [BER 83], les écritures sont considérées comme des créations de nouvelles versions d'objets. Les estampilles sont utilisées pour retourner la version adéquate de l'objet suite à une requête de lecture. Il n'y a donc pas de surcoût de contrôle de concurrence, puisque les différentes versions peuvent être nécessaires à l'algorithme de recouvrement [BER 87]. Dans [KIM 91], Kim et Srivastava ont adapté cet algorithme au contexte temps réel dans le but de réduire le taux de transactions abandonnées. Cependant, cette technique ne peut être appliquée que dans les systèmes où les écritures des données sensorielles (c'est-à-dire des données en provenance de capteurs) correspondent à la création de nouvelles versions pour ces données. Par conséquent, les transactions qui écrivent les valeurs des capteurs ne rentrent pas en

conflit avec celles qui lisent ces valeurs. L'exploitation d'anciennes versions constitue l'obstacle majeur pour utiliser cette technique dans beaucoup d'environnements temps réel où la nécessité d'utiliser des informations fraîches est un critère important.

3.4.3. *Le protocole SCC (Speculative Concurrency Control)*

SCC est un protocole de contrôle de concurrence conçu spécialement pour les SGBD temps réel [BES 94, BRA 95b]. Il est basé sur la *redondance* des ressources. L'objectif est de trouver le plus tôt possible les ordonnancements sérialisables et de les mettre en œuvre, augmentant ainsi les chances pour les transactions de respecter leurs échéances. L'algorithme SCC détecte les conflits éventuels le plus tôt possible et permet à des transactions concurrentes de s'exécuter.

Dans le contexte temps réel et étant donné que les conflits ne sont détectés qu'à la phase de validation avec les protocoles OCC, à partir de quel instant peut-on dire qu'il est trop tard pour redémarrer une transaction ? Le protocole OCC-BC tente de résoudre les problèmes de OCC en utilisant la *notification* qui consiste à ce que la transaction qui effectue une validation avertisse toutes les transactions avec lesquelles elle est en conflit. Ces transactions sont redémarrées immédiatement, augmentant ainsi leurs chances de se terminer avant leurs échéances. Le protocole spéculatif (SCC) franchit une étape de plus dans l'utilisation des connaissances sur les conflits entre transactions. En effet, il « spécule » sur des mesures correctives à prendre dès le début du conflit, en se servant des ressources redondantes. Dès que deux transactions entrent en concurrence sur un granule de donnée, une copie de la transaction qui effectue la lecture est créée. Il faut noter aussi que les deux transactions jumelles peuvent avoir des parcours différents pour arriver à leur validation. En particulier, les conflits qui surviendraient entre chacune d'elles d'une part et les autres transactions du système d'autre part ne seront pas nécessairement les mêmes.

En conclusion, on peut dire que le protocole SCC de base (c'est-à-dire SCC-OB) augmente les chances que les transactions se terminent avant leurs échéances. Cependant, il est très gourmand en ressources, car il est basé sur la duplication des transactions qui s'exécuteraient sur des processeurs différents. Il est donc destiné à des architectures spéciales de systèmes réels. Des variantes ont été développées pour pallier ce problème : SCC-CB, SCC-DC, SCC-kS [BRA 95b], qui sont des algorithmes moins gourmands en ressources. Finalement, on peut considérer le protocole SCC comme un précurseur pour les algorithmes d'ordonnancement des transactions dans les SGBD temps réel. Les applications temps réel étant souvent des applications critiques, l'inconvénient majeur de ces algorithmes, représenté par la nécessité de dupliquer les ressources, est relativement atténué.

3.5. *Etude comparative des différents protocoles*

La plupart des algorithmes de contrôle de concurrence s' appliquent aux transactions à échéances strictes non critiques. Nous récapitulons dans le tableau suivant une analyse effectuée sur les algorithmes présentés précédemment. Nous nous intéressons particulièrement aux points suivants :

– contexte d'utilisation (temps réel strict critique ou non critique, temps réel non strict),
– simplicité de mise en œuvre,
– prise en compte des aspects statiques/dynamiques,
– pré-analyse des transactions,
– duplication des ressources,
– redémarrage/attente des transactions.

Protocoles	Caractéristiques
OCC-BC	– Adapté au temps réel strict non critique, – Simple à mettre en œuvre, – Abandons très nombreux, – Verrouillage des ressources.
Wait-50	– Adapté au temps réel strict non critique, – Simple à mettre en œuvre, – Nombre d'abandons réduit par rapport à OCC-BC.
CCA	– Adapté au temps réel non strict et strict non critique, – Difficile à mettre en œuvre, – Nombre d'abandons très réduit, – Pré-analyse dynamique des transactions.
Abandon par priorité	– Adapté au temps réel strict non critique, – Simple à mettre en œuvre, – Nombre d'abandons réduit, – Abandon au profit de transactions qui ne terminent pas forcément dans les temps.
Héritage de priorités	– Adapté au temps réel strict non critique – Pas d'inversion de priorités, – Durée de blocage imprévisible et blocages en chaîne.
PCP	– Adapté au temps réel strict non critique – N'abandonne pas les transactions, – Borne de temps statique pour la durée de temps de l'inversion de priorité, – Assignation statique des priorités.
Intervalles d'estampilles	– Adapté au temps réel strict non critique, – Diminue les abandons inutiles, – Améliore la détection et la résolution de conflits.
MCC	– Adapté au temps réel strict non critique, – Mise en péril de la cohérence temporelle.
SCC	– Adapté au temps réel strict critique, – Augmente le nombre de transactions qui satisfont leur échéance, – Abandons et redémarrages nombreux, – Nécessité de ressources redondantes.

Tableau 1. *Comparaison des différents protocoles de contrôle de concurrence*

4. Propriétés ACID dans les transactions temps réel

Dans les SGBD traditionnels, tout le monde s'accorde sur les propriétés que doivent satisfaire les transactions pour qu'elles soient correctes. Il s'agit des propriétés ACID [ESW 76]. Dans les SGBD temps réel, ces propriétés doivent être revues pour tenir compte de la sémantique des données temps réel et de la nature des actions dans les applications temps réel [RAM 93, PUR 94]. On parle alors de propriétés partielles (sérialisabilité partielle, …, durabilité partielle), c'est-à-dire, une propriété qui risque d'être insatisfaite temporairement.

a) Isolation partielle : soit par exemple une donnée d ayant une durée de validité d_{dv}. Soit une transaction T de lecture de la donnée d. Si T n'a pas réussi à lire d pendant sa durée de validité, alors dans certaines applications, il est permis à la transaction d'utiliser la version précédente de d (même si sa durée de validité est dépassée), à condition que cette version soit *similaire* à la suivante [KUO 96]. Deux versions d'une donnée sont considérées comme similaires si la transaction qui lit cette donnée les considère comme similaires [KUO 96]. Dans l'exemple présenté en figure 1, à l'instant $t_b = 4$, la transaction lit Y. Son échéance devient $t_f = t_b + 7 = 11$. À l'instant $t_c = 6$, elle lit la donnée Z. Son échéance devient $t_f = t_c + 3 = 9$. Si T ne se termine pas avant t = 9, elle est soit abandonnée, soit autorisée à poursuivre son exécution en utilisant le principe de similarité (si la version suivante de Z est similaire à la précédente). L'idée de similarité est une idée ancienne déjà exploitée dans le domaine de l'avionique par exemple, où les changements de valeurs acquises par des capteurs sont considérés comme négligeables sur un nombre donné de cycles d'acquisition. En utilisant le principe de similarité, plus précisément de la similarité régulière, Kuo et al. [KUO 96] ont introduit un critère plus faible que la sérialisabilité pour l'ordonnancement des transactions temps réel : la Δ–sérialisabilité. D'autres travaux ont défini d'autres critères de correction pour les transactions temps réel en fonction des applications et des objectifs poursuivis. Cependant, dans [KUO 96], les auteurs ont montré que le problème de savoir si un ordonnancement est Δ–sérialisable est NP-Complet. Un protocole de contrôle de concurrence et d'ordonnancement des transactions temps réel basé sur la similarité a été introduit dans [KUO 96] : SSP (Similarity Stack Protocol). L'hypothèse considérée est que la sémantique des applications permet de préciser *la limite de similarité* pour chaque donnée de la base, c'est-à-dire que deux écritures sur un même objet doivent être similaires si leurs estampilles diffèrent d'une quantité inférieure ou égale à la limite de similarité de la donnée. La Δ–sérialisabilité est illustrée par l'exemple suivant, où W(T,X) et R(T,X) représentent respectivement des opérations d'écriture et de lecture de la donnée X, par la transaction T. Soient les ordonnancements O1 et O2 suivants :

Ordonnancement O1 :

```
W(T3,X), R(T1,X), W(T1,X), R(T2,X), R(T2,Y), W(T2,Y), W(T1,Y)
```

Ordonnancement séquentiel (T3, T2, T1) O2 :

```
W(T3,X), R(T2,X), R(T2,Y), W(T2,Y), R(T1,X),  W(T1,X), W(T1,Y)
```

Si `W(T3, X)` et `W(T1, X)` sont des actions d'écriture *similaires*, alors l'ordonnancement O1 est Δ–sérialisable.

Une autre politique d'ordonnancement préconise de retarder la transaction jusqu'à l'occurrence d'une nouvelle version de la donnée à lire. Il s'agit de la politique de retard forcé (Forced Delay [XIO 96a]). Pour l'exemple de la figure 1, si T n'arrive pas à lire Z pendant sa durée de validité, elle est retardée jusqu'à l'occurrence de la version suivante de Z.

D'autres protocoles de contrôle de concurrence sont introduits dans [XIO 96a]. Il s'agit de protocoles classiques EDF (Earliest Deadline First), EDDF (Earliest Data-Deadline First), DDLSF (Data-Deadline based Least Slack First) auxquels est associée la politique d'attente forcée FWE (Forced Wait policy with Estimated Remaining Execution Time). Xiong et al. ont montré à travers des simulations que la politique de retard forcé et la notion de similarité des données utilisées simultanément réduisent les performances des algorithmes d'ordonnancement précédents. Par contre, l'utilisation du retard forcé seul ou de la similarité seule accroît les performances des transactions utilisateurs, notamment si les algorithmes disposent d'informations supplémentaires telles que le temps d'exécution estimé et le temps restant estimé des transactions.

b) Atomicité partielle : L'atomicité d'une transaction est basée sur le principe du "tout ou rien", c'est-à-dire soit toutes les actions de la transaction sont exécutées, soit aucune ne l'est. Étant donné que dans beaucoup d'applications temps réel le respect de l'échéance des transactions est primordial, les SGBD temps réel sur lesquels repose l'application doivent assouplir ce critère et permettre que des transactions dont une partie seulement des actions s'était exécutée d'effectuer leur validation. En effet, il est souvent préférable d'obtenir un résultat partiel (ou imprécis) dans les temps qu'un résultat complet ou précis en retard. Cet assouplissement passe par l'analyse de la sémantique des transactions pour savoir dans quelle mesure la transaction globale pourrait être décomposée. Il serait intéressant, par exemple, d'effectuer une pré-analyse de la transaction [CHA 98] pour dégager plusieurs sous-ensembles d'actions. Des actions qui doivent impérativement respecter leur échéance, des actions que l'on peut abandonner si elles ratent leur échéances et des actions dont l'exécution peut se poursuivre même si la transaction globale a raté son échéance.

c) Cohérence partielle : La cohérence dans une base de données temps réel englobe la cohérence des bases de données traditionnelles (cohérence logique), et une composante temporelle : la cohérence absolue et la cohérence relative [RAM 93]. La cohérence absolue exprime le fait qu'une donnée temps réel doit être utilisée avant la fin de sa durée de validité pour garantir la "fraîcheur" de la base. La cohérence relative concerne les données dérivées à partir de données de base qui doivent être cohérentes entre elles. Dans certaines applications temps réel, il est indispensable que la cohérence temporelle soit respectée au détriment, parfois, de la violation au moins temporaire de la cohérence logique [KIM 94]. Dans ces applications, le fait que la base de données soit, du point de vue interne, incohérente temporairement est moins important si les contraintes temporelles sont respectées. Il s'agit d'applications où les actions possèdent des échéances strictes critiques.

d) Durabilité partielle : Dans les bases de données classiques, la durabilité d'une transaction exprime le fait que le résultat de l'exécution d'une transaction devient permanent une fois que la transaction est validée. Seules des transactions validées peuvent modifier l'état d'une base de données. Le système de stockage d'une base de données en mémoire secondaire garantit de retrouver les effets des transactions validées, même à la suite d'un crash du système. Dans le cas du temps réel, certaines données doivent résider en mémoire principale à cause de leur durée de validité trop courte pour être stockées en mémoire secondaire et être accédées ensuite pour les utiliser. D'autres données temps réel peuvent résider en mémoire secondaire. Dans le premier cas, le stockage obligatoire en mémoire principale exclut la possibilité de garantir la propriété de durabilité. Dans le second cas, comme la durée de vie des données temps réel est limitée, la notion de durabilité des données temps réel peut ne être pas une propriété exigée des transactions temps réel. En effet, il est quasiment impossible que les données temps réel demeurent encore valides à la fin du processus de reprise, suite à une défaillance du SGBD.

On constate donc que les propriétés ACID traditionnelles des transactions sont à assouplir pour pouvoir espérer concevoir et réaliser des SGBD temps réel commerciaux qui respectent les objectifs des applications temps réel.

5. Transactions temps réel à échéances strictes critiques

Les travaux sur le contrôle de concurrence des transactions à échéances strictes critiques et leur ordonnancement sont rares. En effet, beaucoup de chercheurs sur les SGBD temps réel pensent qu'il est impossible d'ordonnancer de telles transactions en maintenant la cohérence temporelle des données, étant donnée la présence de certains facteurs temporels imprévisibles tels que le temps d'accès disque, l'accès à un granule de donnée, les blocages dus à la concurrence entre transactions, etc. Cependant, certains travaux issus de recherche sur l'ordonnancement des tâches temps réel commencent à voir le jour [AUD 93, SON 93, ULU 98, BES 94, BYU 96]. Ces travaux sont basés notamment sur la localisation de la base de données en mémoire principale [COT 98], le surdimensionnement des ressources, la périodicité des transactions strictes critiques et la disponibilité d'estimations des temps d'exécutions des transactions. Dans [SON 93], Son et al. ont développé un prototype de SGBD temps réel de type relationnel gérant des bases de données en mémoire centrale. Il s'agit d'une couche autour du noyau de système d'exploitation temps réel orienté objet, ARTS [TOK 90]. ARTS dispose de mécanismes d'ordonnancement de tâches basés sur les priorités. De conception ouverte, ARTS facilite les extensions, comme l'ordonnancement des transactions du SGBD temps réel, dans lequel l'idée est de pouvoir verrouiller un ou plusieurs granules de données (et non toute une relation de la base). En outre, [SON 93] s'est intéressé aux requêtes sur la base mais n'a pas implanté les opérations relationnelles complexes comme la jointure, jugées inutiles dans un contexte temps réel. Dans [AUD 93], Audsley et al. ont travaillé sur la cohérence temporelle dans les SGBD temps réel stricts critiques. Ils utilisent des mécanismes pour s'assurer que les données sont suffisamment récentes pour garantir la

"fraîcheur" de la base, évitent les blocages en imposant que seul un type de transactions peut accéder à un objet particulier et utilisent des mécanismes non-bloquants d'accès aux données. Le modèle suppose que les applications temps réel manipulent des quantités relativement faibles de données (quelques milliers). Les transactions accèdent à des sous-ensembles bornés de données, connus à l'avance et sont invoquées régulièrement, ce qui permet de trouver de nombreuses possibilités d'ordonnancement moyennant le respect de certaines conditions portant sur les relations qui doivent exister entre la période, l'échéance de la transaction et les intervalles de cohérence absolue des données. Dans [ULU 98], Ulusoy et al. ont défini un protocole de contrôle de concurrence pour les transactions dans les SGBD temps réel gérant des bases de données en mémoire principale sachant qu'une copie de la base non nécessairement récente réside en mémoire secondaire. En éliminant les incertitudes dues aux E/S, ils ont défini des mécanismes pour prévoir le pire cas d'exécution des transactions en effectuant une pré-analyse des transactions. Ils répartissent ainsi les ensembles de relations auxquels les transactions accèdent en ensemble de lectures, ensemble d'écritures, ensemble de conflits, liste des transactions dans la "file des transactions en attente" et liste des transactions dans la "file des transactions prêtes à s'exécuter" [ULU 98]. Dans [BES 94], Bestavros et al. ont défini un protocole de contrôle de concurrence basé sur le surdimensionnement des ressources.

Des ordonnancements tenant compte des contraintes temporelles strictes critiques sont possibles à condition d'imposer certaines restrictions sur les données et les transactions. Davantage de travaux sont donc nécessaires pour explorer toutes les possibilités d'ordonnancement de ce type de transactions en exploitant les nombreux travaux développés sur les tâches temps réel. C'est dans cette direction que nous orientons nos travaux futurs sur les SGBD temps réel.

6. Conclusion

Bien que des travaux de plus en plus nombreux sur les bases de données temps réel commencent à voir le jour, notamment durant les années 90, le travail n’est pas encore suffisamment avancé pour que les industriels commencent à s’intéresser à ce nouvel outil. C’est la conclusion à laquelle ont abouti plusieurs participants à la première rencontre internationale sur les SGBD temps réel [BES 96]. Les participants ont notamment préconisé la réalisation de prototypes pour implanter les différentes fonctionnalités des SGBD temps réel et tester la validité des algorithmes conçus pour l’ordonnancement des transactions temps réel [AND 96, ARA 96]. Ces prototypes doivent montrer la "valeur ajoutée" apportée par les SGBD temps réel par rapport aux SGBD traditionnels pour beaucoup d'applications. D’autre part, beaucoup de problèmes restent ouverts et méritent d’être étudiés :

- le stockage des données temps réel : mémoire principale ou mémoire secondaire, …
- les protocoles de contrôle de concurrence implantables à faible coût,
- la gestion des tampons et l’ordonnancement des opérations d’entrées/sortie,
- les schémas de recouvrement selon les types de données,
- les architectures matérielles pour supporter les SGBD temps réel,

– les méthodes de spécification et de conception d'une base de donnée temps réel, en particulier la manière dont les contraintes temporelles pourraient être exprimées et de quelle façon ces contraintes se répercutent sur les échéances des transactions et des sous-transactions et sur les durées de validité des données [RAM 96b, MAM 98].

Bibliographie

[ABB 88] ABBOT R., GARCIA-MOLINA H., « Scheduling Real-Time Transactions: a performance evaluation », In *Proc. of the 14th Int. Conf. on VLDB*, p. 1-12, 1988.

[AND 96] ANDLER S.F., HANSSON J., ERIKSSON J., MELLON J., BERNDTSSON M., EFTRING B., « DeeDS Towards a Distributed and Active Real-Time Database System », In *ACM SIGMOD Record*, Vol. 15, N° 1, p. 38-40, Mar. 1996.

[ARA 96] ARANHA R.F.M, GANTI V. NARAYANAN S., MUTHUKRISHNAN C.R., PRASAD S.T.S, RAMAMRITHAM K,. « Implementation of a Real-Time Database System », In *Information Systems, Special Issue on Real-Time Databases*, Vol. 21, N° 1, 1996.

[AUD 93] AUDSLEY N., BURNS A., RICHARDSON M., WELLINGS A., « Data Consistency in Hard Real-Time Systems », Tech. Rapp. YCS 203., Univ. of York., U.K. June. 1993.

[BAY 82] BAYER R., ELHARDT K., HEIGERT J., RESEIR A., « Dynamic timestamp allocation for transactions in database systems », In *Proc. of the 2nd Int. Symposium on Distributed Databases*, p. 9-20, Sept. 1982.

[BER 83] BERNSTEIN P., GOODMAN N., « Multiversion concurrency control - theory and algorithms », *ACM Transaction on Database Systems*, Vol. 8, N° 4, p. 465-484, 1983.

[BER 87] BERNSTEIN A., PHILIP A., HADZILACOS V., GOODMAN N., « Concurrency Control And Recovery In Database Systems », Addison-Wesley, 1987.

[BES 94] BESTAVROS A., BRAOUDAKIS S., « Timeliness via Speculation for Real-Time Database », In *Proc. of 14th IEEE RTS Symposium*, Dec. 1994.

[BES 96] BESTAVROS A., LIN K-J, SON S., « Report on First International Workshop on Real-Time Database Systems », In *Proc. of SIGMOD Record*, Vol. 25, N° 3, Sept.1996.

[BOK 87] Boksenbaum C., Cart M., Ferrie J., Pons J., « Concurrent certifications by intervals of timestamps in distributed database systems », In *IEEE Transactions on Software Engineering*, SE-13 (4), p. 409-419, 1987.

[BRA 95a] BRANDING B., BUSHMANN A., « On Providing Soft and Hard Real-Time Capabilities in an Active DBMS », In *Proc. of the Int. Workshop on Active and Real-Time DBS*, p. 158-169, 1995.

[BRA 95b] BRAOUDAKIS S., « Concurrency Control Protocols for Real-Time Databases », *PhD dissertation*, Computer Science Departement, Boston University, 1995.

[BYU 96] BYUN J., WELLING A., BURNS A., « A Worst-Case Behavior Analysis for Hard Real-Time Transactions », *In Proc of the 1st Int. Workshop on RTDBS. Issues and Applications*, California, p 144-149, 1996.

[CHA 98] CHAKRAVARTHY S., HONG D-K., JOHNSON T., « Real-Time Transaction scheduling : A Framwork for Synthesizing Static and Dynamic Factors », In *Journal of RTS*, N° 14, p. 135-170, 1998.

[COT 98] COTTET F., DELACROIX J., KAISER C., MAMMERI Z., « Ordonnancement temps réel. Ordonnancement centralisé », Soumis à Les techniques de l' ingénieur, 1998.

[ESW 76] EWARAN K.P., GRAY J.N., LORIE R.A., TRAIGER I.L., « The Notion of Consistency and Predicate Locks in a Database System », CACM 19, 11, Nov. 1976.

[GRA 92] GRAHAM M.H., « Issues in Real-Time Data Management », In *RTS*, Vol.4, N° 3, p. 185-202, Sept. 1992.

[HAR 90] HARITSA J.R., CAREY M.J., LIVNY M., « Dynamic real-time optimistic concurrency control », In *IEEE Real-Time Systems Symposium*, Florida, p. 94-103, Dec. 1990.

[HAR 92] HARITSA J.R., CAREY M.J., LIVNY M., « Data Access Scheduling in Firm Real-Time Database Systems », In *RTS*, Vol.4, N° 3, p. 203-241, Sept. 1992.

[HUA 91] HUANG J., STANKOVIC J.A., RAMAMRITHAM K., TOWSLEY D., « On using priority inheritance in real-time database », In *Proc. of the 12th RTS Symposium*, Dec. 1991.

[KIM 91] KIM W., SRIVASTAVA J., « Enhancing real-time DBMS performance with multiversion data and priority based disk scheduling », In *Proc. of 12th RTS Symposium*, Dec. 1991.

[KIM 94] KIM Y.K., SON, S.H., « Predictability and Consistency in Real-Time Database Systems », In *Principles of Real-Time Systems*, Ed. Prentice-Hall, 1994, pages. 502-524.

[KUO 96] KUO T.W., MOK A.K., « Real-Time Database - Similarity Semantics and Ressource Scheduling », *ACM SIGMOD Record*, Vol. 25, N° 1, p. 18-22, Mar 1996.

[MAM 85] MAMMERI Z., « Conception d'Applications Réparties en Commande de Processus : une Approche par la Structuration des Données », Thèse de Doctorat en Informatique, Institut National Polytechnique de Lorraine, Nancy, France, 1985.

[MAM 98] MAMMERI Z., « Expression et dérivation des contraintes temporelles dans les applications temps réel », A paraître dans la revue RAIRO JESA, 1998.

[PEN 96] PENG C-S, LIN K-J., « Real-Time Benchmark Design for Avionics Systems », In *Proc. of the 1st Int. Workshop on Real-Time Database Systems. Issues and Applications*, California 1996.

[PUR 94] PURIMETLA B., SIVASANKARAN R.M., RAMAMRITHAM K., STANKOVIC J.A., « Real-Time Databases: Issues and Applications », In Principles of RTS, ed. Printice Hill, 1994.

[RAM 93] RAMAMRITHAM K., « Real-Time Databases », In *Journal of Distributed and Parallel Databases*, Vol. 1, N° 2, p. 199-226, 1993.

[RAM 96a] RAMAMRITHAM K., SIVANSAKARAN R., STANKOVIC J.A., TOWSLEY D.T., XIONG M., « Integrating Temporal, Real-Time, and Active Databases », In *ACM SIGMOD Record. Special Issues on RTDBS*, Vol. 25, N° 1.1996.

[RAM 96b] RAMAMRITHAM K., « Where do Time Constraints Come From and Where do They Go ? », In *Int. Journal of Database Management*, Vol. 7, N° 2, p. 4-10, 1996.

[SON 88] SON S.(ed.), *ACM SIGMOD Record: Special Issue on Real-Time Databases*, 1988.

[SON 92a] SON S.H., LEE J., LIN Y. « Hybrid Protocols Using Dynamic Adjustment of Serialization Order for Real-Time Concurrency Control », In *RTS*, Vol. 4, N° 3, p. 269-276, Sept. 1992.

[SON 92b] SON S.H., LIU J.W.S., « How Well Can Data Temporal Consistency be Maintained ? », In *IEEE Symp. Computer-Aided Control System Design*, 1992.

[SON 93] SON S.H., GEORGE D.W., KIM Y-K., « Developing a Database System for Time-Critical Applications », In *Proc. of IEEE Int. Conf. on Database and Expert System Applications* (DEXA' 93), Prague, Czech, Sept. 1993.

[TOK 90] TOKUDA H., NAKAJIMA T., RAO P., « Real-Time Mach : Toward Predictable Real-Time Systems », In Proc of the USENIX 1990 Mach Worshop, Oct. 1990.

[ULU 98] ULUSSOY O., BUCHMANN A., « A real-time concurrency protocol for main-memory database systems », In *Information Systems*, Vol. 23, N° 2, p.109-125, 1998.

[XIO 96a] XIONG M., STANKOVIC J.A., RAMAMRITHAM K., TOWSLEY D., SIVANSAKARAN R., « Maintaining Temporal Consistency : Issues and algorithms », In *Proc. of the 1st Int. Workshop on RTDBS. Issues and Applications*, p. 1-6, California 1996.

[XIO 96b] XIONG M., SIVANSAKARAN R, STANKOVIC J.A., RAMAMRITHAM K., TOWSLEY D., « Scheduling Transactions with Temporal Constraints : Exploiting Data Semantics », In *Proc. of the 17th IEEE RTS Symposium. Washington*, p. 240-251, Dec. 1996.