データベースシステム 113－45
（1997. 7. 15）

情報放送システム上での並行処理制御方式の考察

白田 由香利*, 飯沢 篤志*
株式会社 次世代情報放送システム研究所
{shirota, izw} @ ibl.co.jp
株式会社リコー 研究開発本部 ソフトウェア研究所
{shirota, izw} @ src.ricoh.co.jp

デジタル放送により大量にかつ広範囲にデータが配信された場合、そのシステム全体には、超多地点のデータベースサイトが含まれる。一般にデータベースシステム上では複数のトランザクションが並行して実行されるが、こうした超大規模分散データベースシステム上では、従来の2相コミットのような強い一貫性管理では効率がよくない。それに代わる並行処理制御として、弱い一貫性を用いた方式が多数提案されているが、本稿では、それらの方式を分類し、我々の対象とするシステムに必要な並行処理制御の要件を考察する。

A Study of Concurrency Control Methods on Information Broadcasting

Yukari Shirota† and Atsushi Iizawa †
Information Broadcasting Laboratories, Inc.
{shirota, izw} @ ibl.co.jp
Software Research Center, R & D Group, Ricoh Company, Ltd.
{shirota, izw} @ src.ricoh.co.jp

If a lot of data are broadcasted to a wide area using digital broadcasted systems, the entire system will in result include a very large number of database sites. In general, multiple transactions concurrently run and the concurrency control for them is required. However, in such a very large distributed database system, the existing tight concurrency control method like a two phase commitment method is not efficient. Therefore, instead of the method, many researchers have proposed weak consistency control methods. In the paper, we survey the weak consistency control and think of the requirements for our target database system.

*株式会社リコーより株式会社次世代情報放送システム研究所に兼任出向中。
†The authors are partly on loan from Ricoh Company, Ltd. to Information Broadcasting Laboratories, Inc.

1 まえがき

近年デジタルテレビ放送技術の発展が顕著であるが、情報システムの放送型配信基盤としてデジタルテレビ放送を活用しようという試みが各種計画されている [1]。デジタルテレビ放送の特徴はデータを速く、安く、大量に送ることができることである。配信されるコンテンツも、従来のようにテレビ番組だけを配信するのではなく、データを放送する機能も統合化されていくと予想される。その統合化されたデジタルテレビ放送を **情報放送**と呼ぶことにする [2]。我々のグループでは、**情報放送**で配信した情報をデータベースに蓄積して、それを有効活用する方式を研究している。デジタル放送により大量にかつ広範囲にデータが配信された場合、そのシステム全体には、超多地点のデータベースサイトが含まれると予想される。一般に、データベースシステム上では複数のトランザクションが並行して実行されるが、そのデータ一貫性を保持するために並行処理制御が必要となる。本稿では、**情報放送**システム上での超大規模分散データベースシステム (DBS) を構築する場合の並行処理制御について考察する。

2 超大規模分散 DBS の特徴

本節では、想定する超大規模分散 DBS に対する要件をあげ、そこから並行処理制御に対する要件を考える。

[要件 1:] データ複製を多地点に作りたい。

想定する超大規模分散 DBS に対する要求仕様として (1) 信頼性の向上、(2) アクセス可能性の向上、(3) 性能の向上、がある。これらの要求を満たすためには、データ複製 (data replication) が効果的である。

[要件 2:] 複製間の一貫性保持は即時に行われなくてもよい。

対象となる DBS では、時間的遅れのある更新 (deferred update) を許す制御方式、つまり一つのトランザクション内だけでシステム全体の一貫性が保証されるのでなく、時間が経過するに従い複製間の間で次第に一貫性がとれていくような制御方式も必要であると考える[1]。

[要件 3:] 一貫性保持のための並行処理制御にかかるコストは少ないほどよい。

一般にデータ複製環境では、サイト数が増えるに従い複製間での一貫性保持に要するコストが増大する、という問題が起こる。昨今の DBMS は殆んど 2 相コミットを実現しているが、こうした強い一貫性を保持する並行処理制御方式では、複製間の一貫性保持とトランザクションの直列可能性は保証される反面、処理コストが増大するため実用には適さない[2]。よって、複製間の一貫性管理は多少ルーズであってもよいから、効率を優先する並行処理制御が望まれる。特にデータが安定している、Write が少なく Read が多いシステムでは「多少データが古くてもよいから速く Read したい」という要望が多い。

[要件 4:] しかし、現在アクセスしようとしているローカルデータが最新版であるか否かを確認する手段はほしい。

[要件 5:] サイトが孤立してもある程度の処理は続行したい。

この要件は、分断された操作 (disconnected operation) が可能であることを意味する。モバイル環境においても分断された操作ができることが望まれているが[3][5]、**情報放送**環境での分断操作には、モバイル環境とは違う要件があるのか否か、検討する必要がある。

以下では、上記 5 つの要件に照らし合わせながら、既存の並行処理制御方式を考察する。

3 並行処理制御方式の分類

既存並行処理制御方式を考察する前に、それらを体系づけるためにどのように分類するか、分類方針を明らかにする。

Gray らは、更新の伝搬方式によってデータ複製モデルを Eager 複製と Lazy 複製の 2 つに分類している [4]。

(1)**Eager 複製**: 全ての複製の更新を一つの原子的なトランザクション内で行うことにより、厳格に全ての複製を同期させる。これにより直列可能性は保証されるが、更新のための効率は下がり、トランザクションに要する時間は長くなる。

(2)**Lazy 複製**: ある複製上で更新トランザクションがコミットされた後、その更新が非同期に他の複製に伝搬していく。効率は向上される反面、古いデータを読む可能性がある。

Eager 複製では、よくロック機構が使われているが、全ての複製に対してロックをかけることは実際難しいので、参加できないサイトがあっても処理が進められることが望まれる。実現方式としては、例えば、以下がある。

分断されているサイト上では、Write は禁止するか、あるいは接続再開時に承認を得るまでいつでもアボート可能な状態にする。接続されているサイト間では、重み

[1] Chundi らはこのような更新の遅れを "さざ波 (ripple)" と称しているが [3]、更新伝搬のイメージをうまく表現していると言えるだろう。

[2] Gray らは、デッドロックの確率及びそれに伴うトランザクションの失敗の確率は、サイト数が 10 倍になると 1000 倍になると論じている [4]。

[3] モバイルの場合、分断された操作が必要な主理由は電池の消耗を防ぐことにある。

つき投票システムを採用し、一定数のグループ内で同期をとる [8, 9, 10, 11][4]。

また、更新をどこで行うかによってデータ複製モデルは次の2つに分けられる。

- 集中方式: 各データごとにプライマリサイトが決められていて、そこでしか更新ができない。他のサイトの複製データは read-only となり、更新のリクエストはプライマリサイトに対して発する。
- 分散方式: どのサイトでもデータ更新が可能。

上記の両方式は eager/lazy のどちらのモデルにも適応可能である。

データ複製モデルよりももっと広い一貫性保持管理方式の分類として、次のような分類がある。

(1) **強い一貫性 (Strong Consistency)**: 一時的なデータ矛盾を認めない。

(2) **弱い一貫性 (Weak Consistency)**: 一時的なデータ矛盾を認める。

強い一貫性を達成しようとすると、共有データの変更を直列化するような操作が必要となり、操作の並列性を低下させることになる。それに対して、弱い一貫性管理では、アプリケーションに応じたセマンティクスを利用した制御を行い一貫性保持にかかるコストを軽減しようとする [6]。弱い一貫性管理の例としては、Cheriton が 1986 年に共有化メモリの分野で発表した、特定のアプリケーション及び問題に特化した problem-oriented shared memory がある [7]。その中で Cheriton は問題に特化した特徴として以下をあげている。

- 使用時になって初めてデータが古くなっていることを検出し、新しいデータに入れ換える。
- 古くなったデータでも十分な正確さがあればよい[5]。
- データは必ずしも必要でない[6]。
- 変更が破棄されてもかまわない。
- 読み出した内容が違っていても投票などの多数決方式で決める。

我々の対象とする超大規模分散 DBS でも、こうした特徴をもつアプリケーションは存在すると考える。こうした問題に特化した意味に依存して、並行処理制御における一貫性保持を緩和することが重要であろう。又、前述した eager 複製モデルは強い一貫性管理に属し、lazy 複製モデルは弱い一貫性管理に分類できる。我々の対象とするシステムでは、強い一貫性と弱い一貫性をアプリケーション内容に応じて使い分けていく必要があると考える。従来のような強い一貫性制御を必要とするアプリケーションは依然として残るだろう。しかし、全体の可用性の向上のために弱い一貫性管理を積極的に取り入れていくことが必要と考える。そこで次節では、今までに発表された弱い一貫性管理方式の特徴について述べる。

4 弱い一貫性管理の方式

以下では弱い一貫性の既存モデルについてその特徴を説明する。

(1) マルチバージョン方式

Lazy 複製モデルでは、直列化可能でない操作を検知するために、**マルチバージョン並行実行制御** (multi-version concurrency control) がよく使われる。以下 MV(MultiVersion) 方式と略記する。MV とは、一つの論理的なデータに対して複数の Read 操作と一つの Write 操作を並行して実行できるという方式である [12]。その内容を以下に記す。

- トランザクション T1 が Write ロックをかけているデータに対して T2 が Read しようとした場合、T2 はそのデータが保持する、以前にコミットされたバージョンを読めばよい。
- T1 が Read ロックをかけているデータに対して T2 が Write をしようとした場合、T1 はそのバージョンを保持する一方、T2 は Write の権利を得る。Write 操作の度に新たなバージョンが作られる。

これをデータ複製環境に適用すると「バージョンが古くてもよいアプリケーションであれば Read だけのトランザクションはローカルに実行できる」ということになる [13, 14]。

Bernstein らは [15, 16] において、MV 方式を整理し、「全ての実行において one-copy 直列化可能性が保証されるならば、その MV 並行実行制御アルゴリズムは正しい」と述べ、その判定のための直列化グラフ (serialization graph) を提示した[7]。

上記の拡張版の MV 方式として**スナップショット分離** (snapshot isolation) 方式がある [17]。この方式は次

[4] 投票定数 (quorum) は以下の条件を満たせば、全サイトが参加しなくても強い一貫性が保持できる。
Read-quorum + Write-quorum > n (n は複製サイト数)
Write-quorum + Write-quorum > n.
これは Read-One-Write-All 方式に比べて効率がよい。

[5] 例えば、銀行データベースの統計情報を計算するには 1 月前の古いデータでも問題ない場合がある。

[6] 例えば、外挿 (補外) 法によりデータを作ったり、あるいはデータを必要としない操作に置き換えたりする場合である。

[7] MV データベースでの複数のトランザクションの実行が one-copy 直列化可能 (one-copy serializable) であるとは、1 バージョン・データベース上で同じトランザクションを逐次的に実行した場合と結果が等しい場合である。

の通り: トランザクション T1 がコミットの準備ができたところで、"コミットタイムスタンプ" を得る。T1 がコミットに成功するのは、「そのコミットタイムスタンプ値が T1 のインターバル ("スタートタイムスタンプ" ～ "コミットタイムスタンプ") 中にあり、かつ T1 が Write したものと同じデータに Write したトランザクション T2」が存在しない場合である。それ以外の場合、T1 はアボートする。T1 がコミットされた時初めてその更新は、T1 の "コミットタイムスタンプ" よりも大きい値の "スタートタイムスタンプ" をもつ全てのトランザクションに公開される。

(2) 差異限定管理 (Bounded Divergence Control) 方式 [18]

複製間の差異をどこまで許すか、という許容範囲を条件として与える方式である。制約としては、時間や値に関するものがある。

N-ignorant トランザクション: Krishnaswamy らが提案する方式は、インクリメンタルに数を増やしていくアプリケーションにおいて、より弱い制約の集合を与えることにより、N 個のトランザクションを独立して並行処理しよう、というものである。例えば、200 人定員の飛行機の座席予約において、210 人までのオーバーブッキングを許す、という弱い制約を与えると、200 人の制約を守る予約トランザクションを最大 11 個まで並行して実行できる。この方式は、並行処理される更新トランザクションの数に対して制約を与えることにより、直列化可能性を緩和するものである。

エプシロン直列可能性 (Epsilon-serializability)[19] を提案した Pu らも、異なった複製を作る更新の数を制御している。

疑似複製 (Quasi-copy)[20, 21]: Alonson らの疑似複製モデルでは、複製するものを限定する選択条件 (selection condition) 及び、複製のオリジナルに対する差異の許容限界を指定する一貫性条件 (coherency condition) の 2 種類がある。例えば株式市場のアプリケーションでは、ユーザは選択条件として「化学薬品メーカの株価属性のみを複製する」を、一貫性条件として「プライマリサイトと自サイトとの値の差異が 5 %までは許す」という条件などが指定できる。以下のように各種の制約が記述可能である。

- 時間的遅れの条件
- バージョンの違いの条件[8]
- リフレッシュする周期の条件
- 値の違いのレベルの条件

その他、時間的差異を制約する方式として、Stonebraker らの Mariposa システム [22] や、Agrawal らの方式 [23] がある。

(3) 意味ベース (Semantic-based) の方式

意味を使うことで弱い一貫性管理を実現する方式。

Orderability: 従来の直列化理論は低レベルの Read と Write 操作に基づいて構成されていた。つまり、オブジェクトのメソッドとしてアプリケーションに公開されるものは Read や Write であったが、オブジェクト指向がデータベースの分野でも活発に議論されるようになって、オブジェクトの意味を用いることによりトランザクションの並列性を増加させる方式が数々開発された。

こうした意味ベースの一貫性制御は、トランザクションの意味に基づく方式 (transaction approach) とオブジェクトの意味に基づくもの (data approach) の 2 つに分類される [24]。

Agrawal らは、一貫性の主張 (assertion) と抽象データ型定義をまとめて直列化に変わる新しい正確さの基準 orderability を提案している [25]。Orderability は、ユーザトランザクションが等しいヒストリの全てのステップにおいて同じ出力値を見ることを主張するものではなく、代わりに、ユーザトランザクションが等しいヒストリにおいて、同じ一貫性主張をもつことを要求する。これにより我々はデータベースやトランザクションの意味的情報を活用することができるようになる。

OSCAR: OSCAR は複製間の弱い一貫性管理のためのアーキテクチャである [26]。このアーキテクチャ中の commutative and associative メソッドでは、途中で一貫性が崩れても、操作が交換法則と結合法則を満たす限り、最終的な結果は一致する。例えば、課金計算のように操作が加算と減算であるアプリケーションに本方式が適応可能である。また、絶対的な値のみを使うトランザクションであれば、他のトランザクションで書き変えられた値を参照する、ということがないので、即、実行できる、と論じている。

5 情報放送上の DBS の要件

情報放送上の超多地点大規模分散 DBS の並行処理制御に対する要件を、どのような特徴に起因するかで以下の 3 つに分類して、考察する。

(1) 大規模分散システムに共通な特徴

信頼性、アクセス可能性、及び性能の向上のため、データ複製を基本とするシステムが適切である。従来、障害

[8] ソフトウェアのパッチレベルが 2 以上違っていてはいけない、というような応用が考えられる。

が発生した場合、ローカルにバックアップをとっておいたデータを元に障害回復を行っていたが、データ複製環境では、近隣サイトからデータを得ることで、バックアップ作業を軽減することが可能となる。つまり、データ複製環境では、あるサイトでデータが破壊されても、全体としてはデータ全体を保持しているので、ローカルなデータ障害は修復可能である。しかし解決すべき課題として以下がある。

＊複製データに対するディレクトリサービス

システムはどこに同じデータが存在するのか知っている必要がある。参照したディレクトリ情報が古くも、問合せにいったとき、そのサイトに望みのデータが無かったとしても、再度他のディレクトリ情報により問合せすれば問題はないだろう。重要な点は、システム全体で識別子をいかに統合化するかである、と考える。

＊障害回復における一貫性保持

近隣サイトからコピーする場合、障害で壊れなかった部分とそのコピーした部分で整合性がとれているか、確認手段が必要である。MV 方式を採用するシステムにおいては、近隣データとローカルサイトのデータのバージョンが異なる場合があり、バージョンが異なるデータを混在させることはサイトとしての一貫性を壊す可能性があるからである。どのような一貫性管理方式をとるにせよ、一貫性管理の実現のためには、システム全体で統一された時計をもつことが重要である。また、システムが巨大化するにつれて、データ伝搬時間が大きくなる傾向があるので、その伝搬時間内に起こったデータ更新をどのように操作するか、という問題もある。

(2) 大規模分散システム上のアプリケーションに共通な特徴

代表的な例として、金銭操作に関するアプリケーションについて考察してみる。銀行口座間の金銭の移動のようなアプリケーションでは、一般に強い一貫性管理が必須である。しかし金銭に関する操作を以下のように分離することにより、各々は交換可能かつ結合法則の成り立つ操作となる。

- 振り込みと引出し (加算・減算)
- 利息及び金利計算 (乗算・割算)

殆んどの課金システムでは、上記の加算・減算の操作だけが行われる。その場合、課金口座の複製を作って独自に操作を行ってもプライマリサイトを決めてそこにデータ更新が伝搬するようにすれば、プライマリサイト上でデータ更新情報は次第に収束に向かう。また、差異限定管理方式の伝搬時間の最大値を設定できるシステムであれば、その伝搬時間を待って、データの利息及び金利計算を行うことが可能になる。それにより非プライマリサイトの一貫性は壊れる可能性があるが、プライマリサイトでは一貫性をもつデータ状況を作り出すことが可能であり、その正しい結果を非プライマリサイトにフィードバックすればよい。金銭に関するアプリケーションにおいて弱い一貫性管理を行うためには前提として、N-ignorant トランザクションのように疑似的な緩い制約を設定できることが必要である。例えば、テレビの受信料のように金額の最大値に上限があり、たとえ徴収に失敗してもその被害限度額が推定できる場合や、銀行口座加入者に対して、一定額をプールしておいてもらうことなどして、制約を緩めることが可能な場合である。

一般に、大規模分散システム上のアプリケーションではアプリケーションの必要とする一貫性管理のレベルに応じて一貫性制約を緩和することが可能であると考える。前述した MV 方式、差異限定管理方式、意味ベースの方式などから、適応可能な方式をアプリケーションの内容に照らして選択するとよいだろう。MV 方式の場合には、そのオプションとして、以下の機能をもつことが望ましい。

- 古いバージョンであるか否かが判断できること。
- one-copy 直列化可能であるか否かを判定できること。
- いつ新しいデータを取得できるのか分かること。

(3) 情報放送アプリケーションに特有な特徴

情報放送システム上にあるデータベース (DB) としては番組／コンテンツ DB、EPG(番組表)DB、利用統計情報 DB[9]、及び課金情報 DB が想定される。それらに対する典型的なトランザクション例としては以下のようなものが考えられる。

- 利用者からの番組検索 (頻度:大)
- 番組制作者の番組新規追加／修正 (頻度:小)
- 番組制作者の EPG 新規追加／修正 (頻度:小)
- 課金情報の収集 (頻度:小だが数は大なので、中)
- 利用統計情報の収集 (頻度:小だが数は大なので、中)

課金情報を除き、殆んどのデータ更新は強い一貫性管理を必要としないと考えられる。情報放送システムとしては、一貫性が緩やかに保たれれば、リアルタイム性はあまり問題としない。それよりは著作権保護、アクセス権管理の問題が重要であると考える。また、一部のデータ

[9]利用者のプライバシー保護の問題があるので、収集できるデータには大きな制限を加えるべきである。

を除き、データ更新の頻度は低いと予想される[10]。課金の操作も銀行口座の操作に比較して、額が比較的少額であること、また、リアルタイム性を問わない[11]こと、などから、弱い一貫性制約が設定しやすいシステムである。

6 まとめ

本稿では、情報放送上での超大規模分散DBSの並行処理として適切な方式は何か、を検討した。結論としては、強い一貫性と弱い一貫性の使い分けが可能なシステムであり、かつ、アプリケーションの内容により柔軟に弱い一貫性保持が行えるシステムであることが望まれる。今後は、実際の仕様としてどのような並行処理制御方式を取捨選択すべきか検討していきたい。

参考文献

[1] 渡辺博則: 「受信端末はテレビかパソコンか」、日経マルチメディア、97.4、pp. 60–63。

[2] 飯沢篤志、浅田一繁、白田由香利: 「情報放送のための超大規模分散データベースシステム」、情報処理学会研究会報告、97-DBS-113-43、1997。

[3] P. Chundi, D. J. Rosenkrantz, and S. S. Ravi: "Deferred Updates and Data Placement in Distributed Databases," *Proc. of 12th Intl. Conf. on Data Engineering*, pp. 469–476, Feb. 1996.

[4] J. Gray, P. Helland, P. O'Neil, and D. Shasha: "The Dangers of Replication and a Solution," *Proc. ACM SIGMOD 96*, pp. 173–182, Montreal Canada, June 1996.

[5] 滝沢誠: 「モバイルデータベースシステム」、信学誌、Vol. 80、No. 4、pp. 331–337、1997。

[6] 前川守、所真理雄、清水謙多郎: 「分散オペレーティングシステム –UNIXの次にくるもの」、共立出版、1991。

[7] D. R. Cheriton: "Problem-Oriented Shared Memory: A Decentralized Approach to Distributed System Design," *Proc. of 6th Intl. Conf. Distributed Computing Systems*, pp. 190–197, 1986.

[8] D. K. Gifford: "Weighted Voting for Replicated Data," *Proc. of 7th ACM Symp. on Operating Systems Principles*, pp. 150–159, 1979.

[9] H. Garcia-Molina and D. Barbará: "How to Assign Votes in a Distributed System," *J. of the ACM*, Vol. 32, No. 4, pp. 841–860, Oct. 1985.

[10] M. Rabinovich and E. D. Lazowska: "Improving Fault Tolerance and Supporting Partial Writes in Structured Coterie Protocols," *Proc. of 1992 ACM SIGMOD Conf. on Management of Data*, pp. 226–235, June 1992.

[10] 例えば、噴火しそうな火山の周囲の観測データや、選挙速報などをリアルタイムで流す番組などが例外として考えられるが、その数は少ないだろう。

[11] 今月の集計に間に合わなければ、次の月の集計で整合性をとればよい。

[11] 白鳥則郎、滝沢誠: 「分散処理」、丸善、1996。

[12] R. Bayer, M. Heller, and A. Reiser: "Parallelism and Recovery in Database Systems," *ACM Trans. on Database Sys.*, Vol. 5, No. 2, pp. 139–156, 1980.

[13] D. Agrawal and S. Sengupta: "Modular Synchronization in Multiversion databases: Version Control and Concurrency Contorl," *Proc. of 1989 ACM SIGMOD Conf. on Management of Data*, pp. 408–417, May 1989.

[14] D. Agrawal and V. Krishnaswamy: "Using Multiversion Data for Non-interfering Execution of Write-only Transactions," *Proc. of 1991 ACM SIGMOD Conf. on Management of Data*, pp. 98–107, May 1991.

[15] P. A. Bernstein and N. Goodman: "Multiversion Concurrency Control – Theory and Algorithms," *ACM Trans. on Database Sys.*, Vol. 8, No. 4, pp. 465–483, Dec. 1983.

[16] P. A. Bernstein, V. Hadzilacos, and N. Goodman: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[17] H. Berenson, P. A. Bernstein, J. Gray, et al.: "A Critique of ANSI SQL Isolation Levels," *Proc. ACM SIGMOD 95*, pp. 1–10, San Jose CA, June 1995.

[18] N. Krishnakumar and A. J. Bernstein: "Bounded Ignorance in Replicted Systems," *Proc. 10th ACM SIGACT-SIGMOD Conf. on Principals of Database Sys.*, pp. 63–74, May 1991.

[19] C. Pu and A. Leff: "Replica Control in Distributed Systems: An Asynchronous Approach," *Proc. of 1991 ACM SIGMOD Conf. on Management of Data*, pp. 377–386, May 1991.

[20] R. Alonson, D. Barbará, and H. Garcia-Molina: "Data Caching Issues in an Information Retrieval System," *ACM Trans. on Database Sys.*, Vol. 15, No. 3, pp. 359–384, Sept. 1990.

[21] R. Alonson and L. L. Cova: "Managing Replicated Copies in Very Large Distributed Systems," *Proc. of Workshop on Management of Replicated Data*, pp. 39–42, Houston, Nov. 1990.

[22] J. Sidell, P. M. Aoki, A. Sah, C. Staelin, M. Stonebraker, and A. Yu: "Data Replication in Mariposa," *Proc. of 12th Intl. Conf. on Data Engineering*, pp. 485–494, Feb. 1996.

[23] D. Agrawal and S. Sengupta: "Modular Synchronization in Distributed, Multiversion databases: Version Control and Concurrency Contorl," *IEEE Trans. on Knowledge and Data Eng.*, Vol. 5, No. 1, pp. 126–137, Feb. 1993.

[24] A. Skarra, S. B. Zdonik: "Concurrency Control and Object-Oriented Databases," *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, Eds., pp. 395–421, ACM Press, New Yourk, 1989.

[25] D. Agrawal, A. E. Addadi, and A. K. Singh: "Consistency and Orderability: Semantic-Based Correctness Criteria for Databases," *ACM Trans. on Database Sys.*, Vol. 18, No. 3, pp. 460–486, Sept. 1993.

[26] A. R. Downing, I. B. Greenberg, and J. M.Peha: "OSCAR: A System for Weak-Consistency Replication," *Proc. of Workshop on Management of Replicated Data*, pp. 26–30, Houston, Nov. 1990.