D-037

仮想DBのバックアップ・リカバリ方式

Backup recovery of Virtual DB

菖蒲 佳右† 渡辺裕太† 和田 雄次†† 澤本潤††† 加藤貴司†††

Keisuke Shobu Yuta Watanabe Yuji Wada Jun Tsukamoto Takashi Kato

1. はじめに

様々なネットワークから大量のデータが収集される現代において，それらのデータに潜在している傾向などをデータマイニングによって発見し，それを意思決定などに役立てることが重要となっている．ユビキタスコンピューティング環境において利用されるデータベース（DB）には，リレーショナルDB，XMLDBなど，多種多様なDB（マルチデータベース）が存在し，分散配置されている．また，同じデータモデルを扱う場合においても，いくつかのDBMSベンダーが存在している．このようなマルチデータベースは大規模かつ大容量であることが多い．それにより，データマイニングを行う分析者はルール発見やそのルールの解釈の作業に集中したいにも関わらず，データマイニングの準備過程であるデータベース選択やデータ収集などの作業に多大な時間を割かねばならなくなり，それがデータ利用上の大きな負担となっている．

2．研究目的・研究概要

１を踏まえ，本研究では分析者の負担を軽減するために，ユビキタスコンピューティング環境に分散配置されているマルチデータベースが，分析者からはあたかも一つのDBMSで管理されているかのように見えるようにするDB仮想化技術を開発する．図１に示すように，例えばリレーショナルDB，XMLDBなどの異種データモデルを扱う際，分析者は仮想化DBMSを介してアクセスする．これにより分析者は，RDBMSやXMLDBMSなどのようなデータモデルの差を意識する必要がなくなり，データの分析などに集中することができる．そのようにして分析者がデータ収集などにかかる負担を軽減し，分析に集中できるようにすることが本研究の目的である．この研究を行うにあたり、障害回復に関する対策も行う必要がある。DBにおいて障害時におけるデータの回復を行うことのできる機能は必須ともいえるものであるが、仮想化DBMSを用いていた場合、障害が発生した際に実DB－仮想DB間おいて不整合が発生してしまう場合がある。各DB単独で障害回復を行ってもデータの回復のみが行われるので、実DB－仮想DBの整合性までは回復できない。そこで、実DB－仮想DB間の整合性を回復するための障害回復機能が必要となる。今回はこの仮想DBにおける障害回復機能について報告する。

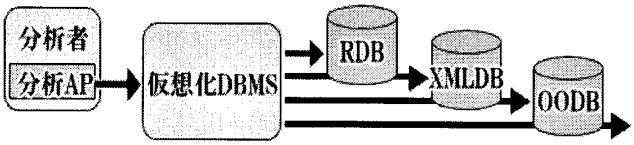

図１．仮想化DBMSの位置づけ

3．DBの仮想化

DBの仮想化には各DBの構造をxmlにより表現したものを用いる。これを共通スキーマと呼ぶ。図２に示すように、RDBによって社員のデータ、XMLDBによって部署のデータを扱うとする。RDBではdatabase1というデータベース内にemployeeテーブルとsalaryテーブルを持ち、それぞれはidとname、idとsalaryというカラムを持つ。XMLDBでは/db/sampledb/というコレクションにpostというキーでXML文書を登録する。

これを共通スキーマで表現すると図３のようになる。dbというノード内にrdbとxmldbというノードが用意される。RDBの構造がrdbに格納され、XMLDBの構造がxmldbに格納される。rdbの子ノードにはmassというノードでデータベース名を、xmldbの子ノードにはcollectionというノードでコレクションを示す。RDBはmassの子にtable、その子にcolumnといった形で表現する。XMLDBはcollectionの子に

†東京電機大学大学院
Tokyo Denki University Graduate School
††東京電機大学
Tokyo Denki University
†††岩手県立大学
Iwate Prefectural University

massというノードを用意し、そのnameにキーを指定する。massの子にはそのXML文書の構造をそのまま示す。ユーザはこのような共通スキーマの情報を基にしたデータ構造から判断し、クエリを発行する。ここでユーザに提示されるデータ構造はRDBやXMLDBといったDBの違いは意識させないようなものにし、共通スキーマ自体は基本的にユーザに提示されない。仮想化DBMSはユーザから要求されたクエリを、共通スキーマを基にして操作対象のDBを判断し、それぞれのDBMSに適したクエリに変換してそれを各DBMSに実行させる。

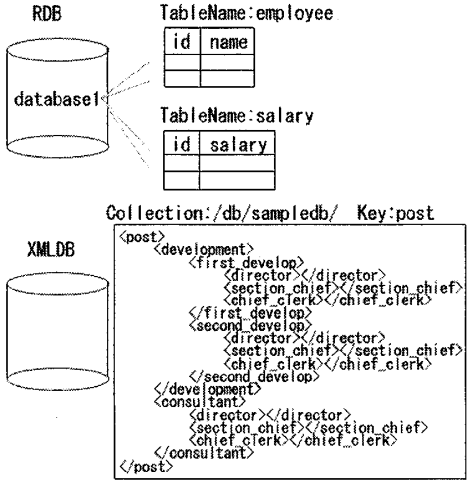

図２．DB例

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<db>
    <rdb>
        <mass name="database1">
            <table name="employee">
                <column>
                    <id></id>
                    <name></name>
                </column>
            </table>
            <table name="salary">
                <column>
                    <id></id>
                    <salary></salary>
                </column>
            </table>
        </mass>
    </rdb>
    <xmldb>
        <collection name="/db/sampledb">
            <mass name="post">
                <station>
                    <development>
                        <first_develop>
                            <director></director>
                            <section_chief></section_chief>
                            <chief_clerk></chief_clerk>
                        </first_develop>
                        <second_develop>
                            <director></director>
                                <section_chief></section_chief>
                            <chief_clerk></chief_clerk>
                        </second_develop>
                    </development>
                    <consultant>
                        <director></director>
                        <section_chief></section_chief>
                            <chief_clerk></chief_clerk>
                    </consultant>
                </station>
            </mass>
        </collection>
    </xmldb>
</db>
```

図３．共通スキーマ例

４．仮想DBのバックアップ・リカバリ

仮想DBにおいてバックアップ・リカバリを行う際、各DBMSに任せた場合の問題点と、それを解消するための仮想化DBMSによる操作について報告する。ここではハードウェア障害が発生したと仮定し、バックアップからデータをリストアし、ログを用いて回復を行う場合について述べる。

4.1 各DBMSの管理による同期問題

仮想化DBMSにバックアップ・リカバリに関する機能を搭載せず、各DBMSにログの取得などを任せた場合にどのようになるかを示す。

仮想化DBMSを用いて図４に示す処理を行ったとする。この例では、id=1の社員についてのデータを【給与：200000⇒250000】、id=2の社員についでのデータを【給与：450000⇒5200000】とし、【post：post.post.consultant.director(8⇒2)】、【post.post.consultant.section_chief(2⇒9)】という更新の命令を行っている。尚、この例ではDB_AがRDB、DB_BがXMLとして扱っており、簡略化のために直列処理としている。もし5番の処理の開始時点で障害が発生したのであれば、DB_Aは処理が中断したと判断できるのでREDOを行わず、DB_Bはそもそも処理が行われていないので回復処理で何もしないことによってDB_AとDB_Bの整合性が保たれることになる。だが、もし12番の処理の開始時点で障害が発生したのであれば、DB_Aとしてはトランザクションが完了しているためにREDOが行われるが、DB_Bでは処理が実行される前の段階であったためにREDOが行われないという状態になる。これにより、仮想DBとしては成立していないトランザクションで実際のDBのデータの一部が変更されているといった不整合が発生してしまう。

仮想化DBMSの処理では、整合性をより保てるように二相コミットメントを用いる。これは各DBMSへすぐにコミット命令を送らず、一度コミットの可否を問い合わせ、全てのDBMSからコミット可能という返答が得られて初めてコミットを行うものである。コミット可否の問い合わせを行った時点での状態をセキュア状態といい、ロールバックとコミットのどちらでも可能な状態となっている。三相コ

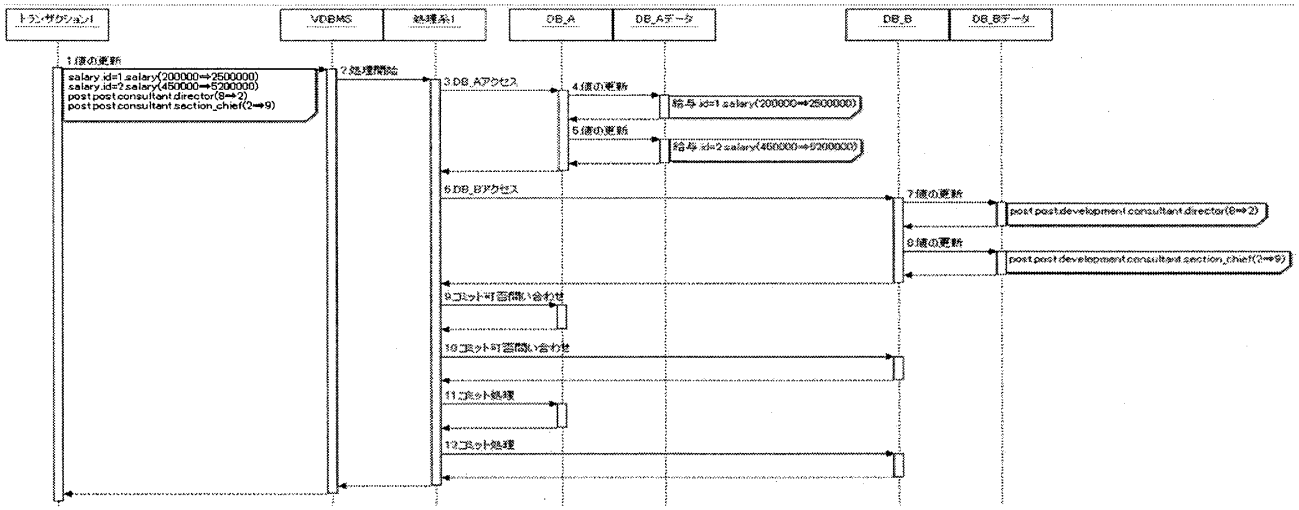

図４．操作例

ミットメントの方がより高いデータの一貫性の保証を得られるが、トランザクションの応答性能が低下するなどのようにパフォーマンスの低下が考えられるので、現時点では二相コミットメントを想定している。

4.2 仮想化DBMSによるバックアップ・リカバリ

4.1で示したように、各DBMSで個別にバックアップ・リカバリを行うようにした場合、障害が発生したタイミングによっては仮想DBと実DBの間で不整合が発生してしまう場合がある。そこで、各DBに対する操作のログを各DBMSではなく仮想化DBMSで管理し、それを基にして障害回復処理を行わせるようにする。仮想化DBMSのログは図５のようにDBの違いを意識させないようなものとし、各要素の先頭にはトランザクションの通し番号を付加する。この通し番号によって並列処理時にログの内容をより明確にする。

```
001 トランザクション開始
001 値の更新(salary.id=1.salary(200000⇒2500000))
001 値の更新(salary.id=2.salary(450000⇒5200000))
001 値の更新(post.post.consultant.director(8⇒2))
001 値の更新(post.post.consultant.section_chief(2⇒9))
001 トランザクション終了
```

図５．仮想化DBMSログ例

ハードウェア障害が発生してしまうとデータが物理的に壊れてしまう為、通常のDBMSと同様に各データベース全体のバックアップを取ることに変わりはない。しかし、仮想DBMSはこれに加えて共通スキーマの情報もバックアップするようにする。この保持した共通スキーマは基本的にはユーザが閲覧することはなく、仮想化DBMSがログに残されている情報を基に各DBのデータを復元する操作を行う際に利用される。まず『トランザクション開始』を検索し、その頭についている番号を調べる。その番号が頭についている『トランザクション終了』が見つかれば障害発生時にコミットされていた処理として回復の対象、見つからなければコミットされていなかった処理ということになり回復の対象にはならない。図５におけるログを基にした場合、まず『001 トランザクション開始』が見つかる。以降を調べると対応する『001 トランザクション終了』も見つけることができ、001のトランザクションは回復対象とみなされる。001にある『値の更新』処理が順に実行されるが、その際にバックアップ処理時に保持しておいた共通スキーマを利用する。『001 値の更新(salary.id=1.salary(200000⇒250000))』となっている場合、（ユーザはデータベースの違いは意識せずにクエリを発行するので）mass以上の階層は一度無視し、massの子に<table name=" salary" >か<salary>があるかを調べる。この例では<rdb><mass name=" database1" >の子に<table name=" salary" >が

見つかるので、RDBであるdatabase1データベースにアクセスする。そして、database1内のsalaryテーブルでid=1となっている行のsalaryの値を200000から250000に変更するためのRDB用のクエリを用意し、それを発行する、という処理となる。また、『001 値の更新(post.post.consultant.director(8⇒2))』となっている場合、massの子に<table name=" post" >か<post>があるかを調べる。この例では<xmldb><collection name="/db/sampledb"><mass key="post">の子に<post>が見つかるので、XMLDB内にある/db/sampledb/コレクションにキーをpostとして登録してあるXML文書にアクセスする。そして、<post><consultant>の子にある<director>の値を8から2に変更するためのXMLDB用のクエリを用意し、それを発行する、という処理となる。尚、『002 トランザクション開始』、『003 トランザクション開始』、『003 トランザクション終了』というログが続いた場合、002のログは無視され、003のログの内容のみがREDOされる。これにより、DB_Aがコミット命令を受け付けてDB_Bへコミット命令を送信する12番の処理の段階で障害が発生した場合にも対応できる。

5．おわりに

今回は以下の三点について述べた。

1．仮想DBはマルチデータベースをユーザからはあたかも1つのDBであるかのように見せることにより、分析者にデータ収集などでかかる負担を軽減する
2．各DBMSの管理によって発生する、既に処理が完了しているDBと処理が中断してしまったDBとの間で発生する回復時の整合性問題
3．仮想化DBMSによるバックアップではデータだけではなく共通スキーマも保持し、それとログを合わせてハードウェア障害におけるリカバリを行う際の手順

4.2ではDB全体のバックアップデータからログと共通スキーマを基にしたハードウェア障害におけるリカバリについて述べたが、システム障害などのようにデータが残っている状態からのリカバリで同様の操作をすると必要以上に時間などのコストがかかってしまう。よって、チェックポイントなどを用いたリカバリの手法を用意する必要があり、今後はその手順の詳細化を行う。チェックポイント処理を行う際には、各DBにおいてチェックポイントのずれが生じる可能性が高く、その点に関する対策が課題となる。また、障害発生のタイミングなどによって様々な障害パターンがあり、仮想DB特有の問題も発生することも考えられ、障害パターンのさらなる洗い出しやそれに対する対策の検討も必要となる。他にも、共通スキーマを基にしてユーザからのクエリを各DBに対応したクエリへ変換する手法の確立、仮想DBのログと実DBのログとの整合性の検証といった課題が残されている。

参考文献

[1]. @IT ＜ http://www.atmarkit.co.jp/ ＞
[2]. IT用語辞典 e-words ＜ http://e-words.jp/ ＞
[3]. 北川博之 著 ： データベースシステム(2005)
[4]. 速水治夫, 宮崎収兄, 山崎晴明 著 ： データベース(2002)
[5]. Serge Abiteboul, Peter Buneman, Dan Suciu 著 ： XMLデータベース入門(2006)
[6]. 水岡祥二, 山之内輝孝, 森脇慎一郎, 片岡信之 著 ： 【データベース】完全教本(2005)