

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号  
特許第7595758号  
(P7595758)

(45)発行日 令和6年12月6日(2024.12.6)

(24)登録日 令和6年11月28日(2024.11.28)

|                          |                |         |  |  |
|--------------------------|----------------|---------|--|--|
| (51)国際特許分類               | F I            |         |  |  |
| G 0 6 F 16/176 (2019.01) | G 0 6 F 16/176 | 1 0 0   |  |  |
| G 0 6 F 9/52 (2006.01)   | G 0 6 F 9/52   | 1 2 0 A |  |  |
| G 0 6 F 9/46 (2006.01)   | G 0 6 F 9/46   | 4 3 0   |  |  |

請求項の数 21 (全59頁)

|                   |                               |          |                      |
|-------------------|-------------------------------|----------|----------------------|
| (21)出願番号          | 特願2023-521163(P2023-521163)   | (73)特許権者 | 517392436            |
| (86)(22)出願日       | 令和4年4月20日(2022.4.20)          |          | ▲騰▼▲訊▼科技(深▲セン▼)有限公   |
| (65)公表番号          | 特表2023-546818(P2023-546818 A) |          | 司                    |
| (43)公表日           | 令和5年11月8日(2023.11.8)          |          | TENCENT TECHNOLOGY   |
| (86)国際出願番号        | PCT/CN2022/087848             |          | (SHENZHEN) COMPANY   |
| (87)国際公開番号        | WO2022/242401                 |          | LIMITED              |
| (87)国際公開日         | 令和4年11月24日(2022.11.24)        |          | 中華人民共和国518057▲広▼▲東   |
| 審査請求日             | 令和5年4月5日(2023.4.5)            |          | ▼省深▲セン▼市南山区高新区科技中一   |
| (31)優先権主張番号       | 202110546080.0                |          | 路 ▲騰▼▲訊▼大厦35▲層▼      |
| (32)優先日           | 令和3年5月19日(2021.5.19)          |          | 35/F, Tencent Build  |
| (33)優先権主張国・地域又は機関 | 中国(CN)                        |          | ing, Kejizhongyi Ro  |
|                   |                               |          | ad, Midwest District |
|                   |                               |          | of Hi-tech Park, Nan |
|                   |                               |          | shan District, Shen  |
|                   |                               |          | zhen, Guangdong 518  |
|                   |                               |          | 最終頁に続く               |

(54)【発明の名称】 データベースシステムのトランザクション処理方法、装置、電子機器、及びコンピュータプログラム

(57)【特許請求の範囲】

【請求項1】

電子機器により実行される、データベースシステムのトランザクション処理方法であって、前記方法は、

ターゲットトランザクションの並列トランザクションを決定するステップであって、前記並列トランザクションと前記ターゲットトランザクションの間には同一変数に作用する読み取り書き込み操作を含み、前記ターゲットトランザクションはコミットすべきトランザクションである、ステップと、

前記ターゲットトランザクションの読み取りセットと前記並列トランザクションの書き込みセットとの間の第1共通集合を取得し、且つ前記ターゲットトランザクションの書き込みセットと前記並列トランザクションの読み取りセットとの間の第2共通集合を取得するステップと、

前記第1共通集合と前記第2共通集合のうちの少なくとも一項が空でないデータセットであり、且つ前記ターゲットトランザクションが前記並列トランザクションと競合するときに、前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定するステップであって、前記ターゲット変数リストは前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数を含む、ステップと、を含むデータベースシステムのトランザクション処理方法。

【請求項2】

前記方法は、

前記ターゲットトランザクションと前記並列トランザクションからロールバックすべきトランザクションを決定するステップと、

前記ロールバックすべきトランザクションに対してロールバックを行うステップと、をさらに含む請求項 1 に記載の方法。

【請求項 3】

前記ターゲットトランザクションと前記並列トランザクションからロールバックすべきトランザクションを決定する前記ステップは、

トランザクション優先度に基づいて、前記ターゲットトランザクションと前記並列トランザクションのうちトランザクション優先度の最も低いトランザクションを前記ロールバックすべきトランザクションとして決定するステップを含む請求項 2 に記載の方法。

10

【請求項 4】

トランザクション優先度に基づいて、前記ターゲットトランザクションと前記並列トランザクションのうちトランザクション優先度の最も低いトランザクションを前記ロールバックすべきトランザクションとして決定する前記ステップの前に、前記方法は、

前記ターゲットトランザクションと前記並列トランザクションにおいて、現在のトランザクションの処理期間がその他のトランザクションの処理期間未満であるときに、前記現在のトランザクションを前記トランザクション優先度の最も低いトランザクションとして決定するステップであって、前記現在のトランザクションは前記ターゲットトランザクションと前記並列トランザクションのうちのいずれか 1 つのトランザクションであり、前記その他のトランザクションは前記ターゲットトランザクションと前記並列トランザクションのうちの前記現在のトランザクション以外のトランザクションである、ステップと、

20

前記ターゲットトランザクションと前記並列トランザクションにおいて、前記現在のトランザクションが読み取りトランザクションであるときに、前記現在のトランザクションを前記トランザクション優先度の最も低いトランザクションとして決定するステップと、

前記ターゲットトランザクションと前記並列トランザクションにおいて、前記現在のトランザクションが書き込みトランザクションであり、且つ前記現在のトランザクションの処理期間が参照期間未満であるときに、前記現在のトランザクションを前記トランザクション優先度の最も低いトランザクションとして決定するステップと、

前記ターゲットトランザクションと前記並列トランザクションにおいて、前記現在のトランザクションが作用した変数数量が前記その他のトランザクションが作用した変数数量よりも大きいときに、前記現在のトランザクションを前記トランザクション優先度の最も低いトランザクションとして決定するステップと、のうちのいずれか 1 つのステップをさらに含む請求項 3 に記載の方法。

30

【請求項 5】

前記ロールバックすべきトランザクションに対してロールバックを行う前記ステップの後に、前記方法は、

ターゲット競合閉路グラフにおける前記ロールバックすべきトランザクションが作用した変数、及び対応する読み取り書き込み操作を削除するステップであって、前記ターゲット競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられ、前記ターゲット競合閉路グラフにおける辺はトランザクション間の関連関係、及び変数間のバージョン変化を表すことに用いられる、ステップをさらに含む請求項 2 ～ 4 のいずれか一項に記載の方法。

40

【請求項 6】

前記方法は、

前記ターゲットトランザクションと前記並列トランザクションとの間の第 1 競合閉路グラフを取得するステップであって、前記第 1 競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子を表すことに用いられ、前記第 1 競合閉路グラフにおける辺はトランザクション間の関連関係、及びトランザクションに対応する読み取り

50

書き込み操作が作用した変数を表すことに用いられる、ステップと、

前記第 1 競合閉路グラフに対して変換を行い、第 2 競合閉路グラフを得るステップであって、前記第 2 競合閉路グラフにおける頂点は複数の異なるバージョンの変数を表すことに用いられ、前記第 2 競合閉路グラフにおける辺は変数間のバージョン変化、及び同一トランザクションが作用した変数を表すことに用いられる、ステップと、

前記第 2 競合閉路グラフに対して変換を行い、前記ターゲット競合閉路グラフを得るステップと、をさらに含む請求項 5 に記載の方法。

【請求項 7】

前記第 1 競合閉路グラフに対して変換を行い、第 2 競合閉路グラフを得る前記ステップは、

前記第 1 競合閉路グラフの辺に対して分割を行い、前記複数の異なるバージョンの変数を得るステップと、

前記複数の異なるバージョンの変数を頂点として、隣接するバージョンの同一変数に対応する頂点の間に辺を追加し、且つ同一トランザクションが作用した変数に対応する頂点の間に辺を追加して、前記第 2 競合閉路グラフを得るステップと、を含む請求項 6 に記載の方法。

【請求項 8】

前記第 2 競合閉路グラフに対して変換を行い、前記ターゲット競合閉路グラフを得る前記ステップは、

前記第 2 競合閉路グラフにおける同一トランザクションが作用した変数に対応する頂点を合併し、且つ同一トランザクションに対応するトランザクション識別子を合併後の頂点として決定するステップと、

合併後の前記第 2 競合閉路グラフの隣接する頂点の間に辺を追加して、第 3 競合閉路グラフを得るステップと、

前記第 3 競合閉路グラフの 2 つの隣接する頂点中に同一バージョンの変数を含むときに、前記 2 つの隣接する頂点を合併し、且つ合併後の頂点における同一バージョンの変数を消去して、前記ターゲット競合閉路グラフを得るステップと、

前記第 3 競合閉路グラフの 2 つの隣接する頂点中に異なるバージョンの同一変数を含むときに、前記 2 つの隣接する頂点を合併し、前記ターゲット競合閉路グラフを得るステップと、を含む請求項 6 に記載の方法。

【請求項 9】

前記方法は、

前記第 1 共通集合が前記空でないデータセットであるときに、複数のトランザクションで構成された競合整合性状態グラフの、前記ターゲットトランザクションと前記並列トランザクションとの間の有向辺の数量に対して増分処理を行い、且つ前記第 1 共通集合における変数を前記ターゲット変数リスト中に追加し、且つ前記第 1 共通集合中にバージョンが異なる変数を含むときに、前記バージョン変更回数に対して増分処理を行うステップであって、前記有向辺の数量は前記ターゲットトランザクションと前記並列トランザクションとの間が競合するか否かを指示することに用いられる、ステップと、

前記第 2 共通集合が前記空でないデータセットであるときに、前記有向辺の数量に対して増分処理を行い、且つ前記第 2 共通集合における変数を前記ターゲット変数リスト中に追加し、且つ前記第 2 共通集合中にバージョンが異なる変数を含む場合、前記バージョン変更回数に対して増分処理を行うステップと、

第 3 共通集合が前記空でないデータセットであるときに、前記並列トランザクションのコミット状態に基づいて、前記データ異常タイプを決定するステップであって、前記第 3 共通集合は前記ターゲットトランザクションの書き込みセットと前記並列トランザクションの書き込みセットとの間の共通集合である、ステップと、をさらに含む請求項 1 ～ 4 のいずれか一項に記載の方法。

【請求項 10】

前記第 1 共通集合と前記第 2 共通集合のうちの少なくとも一項が空でないデータセット

10

20

30

40

50

であり、且つ前記ターゲットトランザクションが前記並列トランザクションと競合するときに、前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定する前記ステップは、

前記ターゲットトランザクションと前記並列トランザクションにおいて同一変数に作用するターゲット読み取り書き込み操作に基づいて、前記データ異常タイプの所属カテゴリを決定するステップと、

前記所属カテゴリ、及び前記ターゲット変数リストに基づいて、前記所属カテゴリにおけるサブカテゴリを決定するステップと、

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定するステップと、を含む請求項 1～4 のいずれか一項に記載の方法。

#### 【請求項 11】

前記ターゲットトランザクションと前記並列トランザクションにおいて同一変数に作用するターゲット読み取り書き込み操作に基づいて、所属カテゴリを決定する前記ステップは、

前記ターゲットトランザクションと前記並列トランザクションにおける前記ターゲット読み取り書き込み操作が書き込み異常条件を満たすときに、前記所属カテゴリが書き込み異常カテゴリであると決定するステップであって、前記書き込み異常条件を満たすとは、前記ターゲット読み取り書き込み操作が異なるバージョンの同一変数に作用する第 1 指定数量の書き込み操作を含むことを指す、ステップと、

前記ターゲット読み取り書き込み操作が前記書き込み異常条件を満たさず、且つ読み取り異常条件を満たすときに、前記所属カテゴリが読み取り異常カテゴリであると決定するステップであって、前記読み取り異常条件を満たすとは、前記ターゲット読み取り書き込み操作が同一バージョンの同一変数に作用する読み取り操作と書き込み操作を含むことを指す、ステップと、

前記ターゲット読み取り書き込み操作が前記書き込み異常条件と前記読み取り異常条件を満たさず、且つ交差異常条件を満たすときに、前記所属カテゴリが交差異常カテゴリであると決定するステップであって、前記交差異常条件を満たすとは、前記ターゲット読み取り書き込み操作が異なるバージョンの同一変数に作用する読み取り操作と書き込み操作を含むことを指す、ステップと、のうちのいずれか 1 つのステップを含む請求項 10 に記載の方法。

#### 【請求項 12】

前記所属カテゴリ、及び前記ターゲット変数リストに基づいて、前記所属カテゴリにおけるサブカテゴリを決定する前記ステップは、

前記ターゲット変数リストにおける変数数量が第 2 指定数量であるときに、前記所属カテゴリにおける前記サブカテゴリがシングルメタデータ異常であると決定するステップと、

前記ターゲット変数リストにおける変数数量が第 3 指定数量であるときに、前記所属カテゴリにおける前記サブカテゴリがダブルメタデータ異常であると決定するステップと、

前記ターゲット変数リストにおける変数数量が第 4 指定数量であるときに、前記所属カテゴリにおける前記サブカテゴリがマルチメタデータ異常であると決定するステップであって、前記第 4 指定数量は、前記第 2 指定数量、及び前記第 3 指定数量のいずれとも異なる、ステップと、のうちのいずれか 1 つのステップを含む請求項 11 に記載の方法。

#### 【請求項 13】

前記所属カテゴリが前記書き込み異常カテゴリであり、且つ前記サブカテゴリが前記シングルメタデータ異常であるときに、

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

前記バージョン変更回数が第 2 指定数量であるときに、前記書き込み異常カテゴリ、及

10

20

30

40

50

び前記シングルメタデータ異常に基づいて、前記データ異常タイプが紛失自己更新異常であると決定するステップと、

前記バージョン変更回数が第3指定数量であり、且つ前記ターゲット読み取り書き込み操作が第5指定数量の書き込み操作であるときに、前記書き込み異常カテゴリ、及び前記シングルメタデータ異常に基づいて、前記データ異常タイプが全書き込み異常であると決定するステップと、

前記バージョン変更回数が前記第3指定数量であり、且つ前記ターゲット読み取り書き込み操作が第1指定数量の書き込み操作と第6指定数量の読み取り操作であるときに、前記書き込み異常カテゴリ、及び前記シングルメタデータ異常に基づいて、前記データ異常タイプが紛失更新異常であると決定するステップと、のうちのいずれか1つのステップを含む請求項12に記載の方法。

10

【請求項14】

前記所属カテゴリが前記書き込み異常カテゴリであり、且つ前記サブカテゴリが前記ダブルメタデータ異常であり、及び前記バージョン変更回数が第2指定数量であるときに、

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

前記ターゲット読み取り書き込み操作が第1条件と第2条件を満たすときに、前記書き込み異常カテゴリ、前記ダブルメタデータ異常、及び前記第2指定数量に基づいて、前記データ異常タイプが読み取り書き込み半順序異常であると決定するステップであって、前記第1条件を満たすとは、前記ターゲット読み取り書き込み操作が異なるバージョンの第1変数に作用する書き込み操作と読み取り操作を含むことを指し、前記第2条件を満たすとは、前記ターゲット読み取り書き込み操作が異なるバージョンの第2変数に作用する第1指定数量の書き込み操作を含むことを指す、ステップと、

20

前記ターゲット読み取り書き込み操作が第3条件と前記第2条件を満たすときに、前記書き込み異常カテゴリ、前記ダブルメタデータ異常、及び前記第2指定数量に基づいて、前記データ異常タイプがリピータブルライト半順序異常であると決定するステップであって、前記第3条件を満たすとは、前記ターゲット読み取り書き込み操作が同一バージョンの前記第1変数に作用する書き込み操作と読み取り操作を含むことを指す、ステップと、

前記ターゲット読み取り書き込み操作が第4条件と前記第2条件を満たすときに、前記書き込み異常カテゴリ、前記ダブルメタデータ異常、及び前記第2指定数量に基づいて、前記データ異常タイプが全書き込み半順序異常であると決定するステップであって、前記第4条件を満たすとは、前記ターゲット読み取り書き込み操作が異なるバージョンの前記第1変数に作用する前記第1指定数量の書き込み操作を含むことを指す、ステップと、のうちのいずれか1つのステップを含む請求項12に記載の方法。

30

【請求項15】

前記所属カテゴリが前記読み取り異常カテゴリであり、且つ前記サブカテゴリが前記シングルメタデータ異常であり、及び前記バージョン変更回数が第2指定数量であるときに、

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

40

前記ターゲット読み取り書き込み操作が同一変数に作用する第1指定数量の数の読み取り操作と第6指定数量の書き込み操作を含む場合、前記読み取り異常カテゴリ、前記シングルメタデータ異常、及び前記第2指定数量に基づいて、前記データ異常タイプがノンリピータブルリード異常であると決定するステップと、

前記ターゲット読み取り書き込み操作が同一変数に作用する前記第1指定数量の書き込み操作と前記第6指定数量の読み取り操作を含む場合、前記読み取り異常カテゴリ、前記シングルメタデータ異常、及び前記第2指定数量に基づいて、前記データ異常タイプが中間読み取り異常であると決定するステップと、のうちのいずれか1つのステップを含む請求項12に記載の方法。

50

## 【請求項 16】

前記所属カテゴリが前記読み取り異常カテゴリであり、且つ前記サブカテゴリが前記ダブルメタデータ異常であるときに、

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

前記バージョン変更回数が第7指定数量であるときに、前記読み取り異常カテゴリと前記ダブルメタデータ異常に基づいて、前記データ異常タイプが書き込み読み取り半順序異常であると決定するステップと、

前記バージョン変更回数が第2指定数量であるときに、前記読み取り異常カテゴリと前記ダブルメタデータ異常に基づいて、前記データ異常タイプが読み取り半順序異常であると決定するステップと、のうちのいずれか1つのステップを含む請求項12に記載の方法。

10

## 【請求項 17】

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

前記所属カテゴリが前記交差異常カテゴリであり、且つ前記サブカテゴリが前記ダブルメタデータ異常であり、且つ前記バージョン変更回数が第2指定数量であるときに、前記データ異常タイプが書き込み半順序異常であると決定するステップを含む請求項12に記載の方法。

20

## 【請求項 18】

前記所属カテゴリにおける前記サブカテゴリ、前記バージョン変更回数、及び前記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、前記データ異常タイプを決定する前記ステップは、

前記所属カテゴリが前記書き込み異常カテゴリであり、且つ前記サブカテゴリが前記マルチメタデータ異常であるときに、前記データ異常タイプが階段式書き込み異常であると決定するステップと、

前記所属カテゴリが前記読み取り異常カテゴリであり、且つ前記サブカテゴリが前記マルチメタデータ異常であるときに、前記データ異常タイプが階段式読み取り異常であると決定するステップと、

30

前記所属カテゴリが前記交差異常カテゴリであり、且つ前記サブカテゴリが前記マルチメタデータ異常であるときに、前記データ異常タイプが階段式交差異常であると決定するステップと、のうちのいずれか1つのステップを含む請求項12に記載の方法。

## 【請求項 19】

トランザクション処理装置であって、前記装置は、

ターゲットトランザクションの並列トランザクションを決定するように配置される第1決定モジュールであって、前記並列トランザクションと前記ターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含み、前記ターゲットトランザクションはコミットすべきトランザクションである、第1決定モジュールと、

前記ターゲットトランザクションの読み取りセットと前記並列トランザクションの書き込みセットとの間の第1共通集合を取得し、且つ前記ターゲットトランザクションの書き込みセットと前記並列トランザクションの読み取りセットとの間の第2共通集合を取得し、前記第1共通集合と前記第2共通集合のうちの少なくとも一項が空でないデータセットであり、且つ前記ターゲットトランザクションが前記並列トランザクションと競合するときに、前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定するように配置される第2決定モジュールであって、前記ターゲット変数リストは前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数を含む、第2決定モジュールと、を含むトランザクション処理装置。

40

## 【請求項 20】

50

電子機器であって、前記電子機器はプロセッサとメモリとを含み、前記メモリは少なくとも1つのコンピュータプログラムを記憶することに用いられ、前記少なくとも1つのコンピュータプログラムは前記プロセッサによりロードされ且つ実行されるときに、請求項1～4のいずれか一項に記載のデータベースシステムのトランザクション処理方法を実現する電子機器。

【請求項21】

コンピュータプログラムであって、プロセッサにより実行されるときに、請求項1～4のいずれか一項に記載のデータベースシステムのトランザクション処理方法を実現するコンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本願はデータベースの技術分野に関し、データベースシステムのトランザクション処理方法、装置、電子機器、コンピュータ可読記憶媒体、及びコンピュータプログラム製品に関する。

【0002】

本願は出願番号が第202110546080.0号であり、出願日が2021年5月19日である中国特許出願に基づいて提案され、且つ該中国特許出願の優先権を主張し、該中国特許出願の全内容はここで本願に参考として組み込まれている。

【背景技術】

【0003】

データベース技術の発展に伴って、データベースシステムにおいて如何にデータ異常を識別するかが1つの重要な問題となっている。現状では、2種類のデータ異常識別方法があり、その1つは、ロッキング技術を利用して、ロックの排他メカニズムを介して、データ異常を識別する方法である。もう1つは依存グラフ技術を利用し、並列トランザクションにより構成された依存グラフにおいて閉路が存在するか否かを確認し、もし閉路が存在すれば、それはデータ異常が存在するということを説明しているという方法である。しかしながら、ロッキング技術はデータベースシステムの並列性を制限してトランザクション処理効率の低下を引き起こし、また、依存グラフ技術は各並列トランザクションをトラバースルして閉路の存在を識別する必要があるため、トランザクション処理効率が比較的低くなることを引き起こす。

【発明の概要】

【課題を解決するための手段】

【0004】

本願の実施例はデータベースシステムのトランザクション処理方法、装置、電子機器、コンピュータ可読記憶媒体、及びコンピュータプログラム製品を提供しており、データベースシステムのトランザクション処理効率を向上させることを可能にする。

【0005】

本願の実施例はデータベースシステムのトランザクション処理方法を提供しており、該方法は、

ターゲットトランザクションの並列トランザクションを決定するステップであって、前記並列トランザクションと前記ターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含み、前記ターゲットトランザクションはコミットすべきトランザクションである、ステップと、

前記ターゲットトランザクションの読み取りセットと前記並列トランザクションの書き込みセットとの間の第1共通集合を取得し、且つ前記ターゲットトランザクションの書き込みセットと前記並列トランザクションの読み取りセットとの間の第2共通集合を取得するステップと、

前記第1共通集合と前記第2共通集合のうちの少なくとも一項が空でないデータセットであり、且つ前記ターゲットトランザクションが前記並列トランザクションと競合すると

10

20

30

40

50

きに、前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定するステップであって、前記ターゲット変数リストは前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数を含む、ステップと、を含む。

【0006】

本願の実施例はトランザクション処理装置を提供しており、該装置は、

ターゲットトランザクションの並列トランザクションを決定するように配置される第1決定モジュールであって、前記並列トランザクションと前記ターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含み、前記ターゲットトランザクションはコミットすべきトランザクションである、第1決定モジュールと、

前記ターゲットトランザクションの読み取りセットと前記並列トランザクションの書き込みセットとの間の第1共通集合を取得し、且つ前記ターゲットトランザクションの書き込みセットと前記並列トランザクションの読み取りセットとの間の第2共通集合を取得し、前記第1共通集合と前記第2共通集合のうちの少なくとも一項が空でないデータセットであり、且つ前記ターゲットトランザクションが前記並列トランザクションと競合するときに、前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定するように配置される第2決定モジュールであって、前記ターゲット変数リストは前記ターゲットトランザクションと前記並列トランザクションが作用した同一変数を含む、第2決定モジュールと、を含む。

【0007】

本願の実施例は電子機器を提供しており、該電子機器はプロセッサと、メモリとを含み、該メモリは少なくとも1つのコンピュータプログラムを記憶することに用いられ、該少なくとも1セグメントのコンピュータプログラムは該プロセッサによりロードされ且つ実行されるときに、本願の実施例により提供されたデータベースシステムのトランザクション処理方法を実現する。

【0008】

本願の実施例はコンピュータ可読記憶媒体を提供しており、該コンピュータ可読記憶媒体において少なくとも1つのコンピュータプログラムが記憶されており、該少なくとも1つのコンピュータプログラムはプロセッサによりロードされ且つ実行されるときに、本願の実施例により提供されたデータベースシステムのトランザクション処理方法を実現する。

【0009】

本願の実施例はコンピュータプログラム製品を提供しており、該コンピュータプログラム製品はコンピュータプログラム、又は指令を含み、該コンピュータプログラム、又は指令はプロセッサに実行されるときに、本願の実施例により提供されたデータベースシステムのトランザクション処理方法を実現する。

【0010】

本願の実施例により提供された技術的解決手段による有益な効果は、少なくとも以下を含む。ターゲットトランザクションの並列トランザクションを決定した後に、該ターゲットトランザクションの読み取りセットと該並列トランザクションの書き込みセットとの間の第1共通集合、及び該ターゲットトランザクションの書き込みセットと該並列トランザクションの読み取りセットとの間の第2共通集合のうち空でないデータセットが存在する場合に、ターゲットトランザクションが並列トランザクションと競合するか否かを判断し、もしターゲットトランザクションが並列トランザクションと競合すれば、この2つのトランザクションが作用した同じ変数、及び同じ変数のバージョン変更回数に従って、データ異常タイプを決定する。このようにすると、ターゲットトランザクションの実行過程において、ターゲットトランザクションが並列トランザクションと本格的に競合を構成するか否かを確認することを可能にし、また、2つのトランザクションが作用した同じ変数によりデータ異常タイプを決定することで、データベースシステムにおける各種のデータ異常を識別して、データ状態の整合性を保証することを可能にし、それによりデータベース

システムのトランザクション処理効率を向上させる。

【0011】

本願の実施例における技術的解決手段をより明確に説明するために、以下、本願の実施例の記述において使用される必要がある図面を簡単に紹介する。明らかなように、以下の記述における図面は単に本願のいくつかの実施例であり、当業者にとっては、創造的な労働を必要としない前提において、さらにこれらの図面に従ってその他の図面を取得することができる。

【図面の簡単な説明】

【0012】

【図1】本願の実施例に従って提供されたトランザクション処理方法の実施環境の模式図である。

10

【図2】本願の実施例に従って提供された競合閉路グラフの模式図である。

【図3】本願の実施例に従って提供された他の競合閉路グラフの模式図である。

【図4】本願の実施例に従って提供された別の競合閉路グラフの模式図である。

【図5】本願の実施例に従って提供されたトランザクション処理方法のフローチャートである。

【図6】本願の実施例に従って提供された他のトランザクション処理方法のフローチャートである。

【図7】本願の実施例に従って提供されたシングルメタデータ異常CCGグラフの模式図である。

20

【図8】本願の実施例に従って提供されたダブルメタデータ異常CCGグラフの模式図である。

【図9】本願の実施例に従って提供されたマルチメタデータ異常CCGグラフの模式図である。

【図10】本願の実施例に従って提供されたターゲット競合閉路グラフを取得する模式図である。

【図11】本願の実施例に従って提供されたトランザクション処理装置の構造模式図である。

【図12】本願の実施例に従って提供されたサーバの構造模式図である。

【発明を実施するための形態】

30

【0013】

本願の目的、技術的解決手段、及び利点をより明確にするために、以下、図面を組み合わせる本願の実施形態をさらに詳細に記述する。

【0014】

ここでは、例示的な実施例に対して詳細な説明を行い、その例示は図面において示される。以下の記述は図面に関するときに、特に断りのない限り、異なる図面における同じ数字は同じ、又は類似する要素を表す。以下の例示的な実施例において記述された実施形態は、本願と整合するすべての実施形態を代表するものではない。逆に、それらは添付した特許請求の範囲において詳しく記述された、本願のいくつかの態様と整合する装置、及び方法の例に過ぎない。

40

【0015】

本願における用語「第1」「第2」等の文字は作用、及び機能が基本的に同じである同じ項目、又は類似項目に対して区別を行うことに用いられる。理解すべき点としては、「第1」、「第2」、…、「第7」の間は論理的、又は時系列的な依存関係を有するものではなく、数量、及び実行順序に対しても限定を行うものではない。さらに理解すべき点としては、以下の記述では、第1、第2等の用語を使用して各種の要素を記述するが、これらの要素は用語に制限されるべきではない。

【0016】

これらの用語は1つの要素を他の要素と区別することに用いられるのみである。たとえば、各種の例示の範囲から逸脱しない場合において、第1トランザクションは第2トラン

50

ザクションと呼称されることが可能になり、且つ同様に、第2トランザクションも第1トランザクションと呼称されることが可能になる。第1トランザクションと第2トランザクションはいずれもトランザクションであってもよく、且つあるいくつかの場合に、単独であり、且つ異なるトランザクションであってもよい。

#### 【0017】

ここで、少なくとも1つとは、1つ、又は1つ以上であることを指し、たとえば、少なくとも1つのトランザクションは、1つのトランザクション、2つのトランザクション、3つのトランザクション等の1以上の任意の整数個のトランザクションであってもよい。複数とは、2つ、又は2つ以上であることを指し、たとえば、複数のトランザクションは、2つのトランザクション、3つのトランザクション等の2以上の任意の整数個のトランザクションであってもよい。

10

#### 【0018】

本願の実施例を紹介する前に、いくつかのクラウド技術分野における基本概念を導入する必要がある。

#### 【0019】

クラウド技術 (Cloud Technology) : ワイドエリアネットワーク、又はローカルエリアネットワークにおいてハードウェア、ソフトウェア、ネットワーク等の一連のリソースを統一して、データの計算、記憶、処理、及び共有を実現する一種のホスティング技術を指す。すなわち、クラウドコンピューティングビジネスモデルの応用に基づくネットワーク技術、情報技術、統合技術、管理プラットフォーム技術、及びアプリケーション技術等の総称であり、リソースプールを構成することができ、必要に応じて使用することができ、柔軟で便利である。本願の実施例が提供したデータベースシステムのトランザクション処理方法 (以下、トランザクション処理方法と略称する) は、クラウド技術に基づいて実現されてもよい。

20

#### 【0020】

クラウド記憶 (Cloud Storage) : クラウドコンピューティングの概念から拡張して発展した1つの新しい概念であり、分散型クラウド記憶システム (以下、記憶システムと略称する) とは、クラスターアプリケーション、グリッド技術、及び分散記憶ファイルシステム等の機能により、ネットワークにおける大量の各種の異なるタイプの記憶機器 (記憶機器は記憶ノードとも呼ばれる) をアプリケーションソフトウェア、又はアプリケーションインタフェースにより集合して協働させ、外部ヘデータ記憶、及びビジネスアクセス機能を共同で提供する1つの記憶システムを指す。本願の実施例が提供したトランザクション処理方法は、クラウド記憶において応用することができ、それによりクラウド記憶における異常に対して識別を行う。

30

#### 【0021】

データベースシステム (Database) : 一種の電子化されたファイルキャビネットであり、すなわち電子ファイルを記憶する場所でもあり、ユーザーはファイルにおけるデータに対して新規追加、問い合わせ、更新、及び削除等の操作を行うことができる。いわゆる「データベースシステム」とは、所定の方式で共に記憶され、複数のユーザーと共有することができ、できるだけ小さな冗長度を有し、アプリケーションプログラムとは互いに独立したデータセットである。本願の実施例が提供したトランザクション処理方法は、データベースシステムにおいて応用することができ、それによりデータベースシステムにおける異常を識別する。

40

#### 【0022】

本願の実施例に関するデータベースシステムは、スタンドアロンデータベースシステム、ランザクションを主とするスタンドアロントデータベースシステム、分析型を主とするがトランザクション処理能力を必要とするスタンドアロンデータベースシステムであってもよく、一般的な非リレーショナルデータベース (Non-relational SQL、NoSQL) 体系を指してもよく、また、分散型データベースシステム、分散型ビッグデータ処理システムであってもよい。分散型データベースシステムについて言えば、異

50

なる変数が異なる物理ノード上に分散して記憶されるときに、データ状態整合性モデル中に2つ及び2つ以上の変数が存在する場合と対応しており、データ状態整合性モデル、及び相応なトランザクション処理プロセスはいずれも後続の記述において詳細に説明される。

#### 【0023】

データベースシステムにおいて少なくとも1つのノード機器を含むことができ、各ノード機器のデータベースにおいて複数のデータテーブルが記憶されていることができ、各データテーブルは1つ、又は複数のデータ項目（変数バージョンとも呼称される）を記憶することに用いることができる。ここで、ノード機器のデータベースはいずれか1つのタイプの分散型データベースであってもよく、リレーショナルデータベース、又は非リレーショナルデータベースのうちの少なくとも一項を含むことができ、たとえば構造化問い合わせ言語（Structured Query Language、SQL）データベース、NoSQL、一般的な各種の新規の拡張可能／高性能データベース（NewSQL）等であり、本願の実施例においてデータベースのタイプに対しては限定しない。

10

#### 【0024】

いくつかの実施例においては、さらに一種のブロックチェーン技術に基づくデータベースシステム（以下、「ブロックチェーンシステム」と略称する）に応用することができる。上記ブロックチェーンシステムは、本質的に一種の非集中化された分散型データベースシステムに属し、コンセンサスアルゴリズムを採用してブロックチェーン上の異なるノード機器に記載された台帳データの整合を保持し、暗号アルゴリズムにより異なるノード機器間の台帳データの暗号化伝送、及び改竄不可能を保証し、スクリプトシステムにより台帳機能を拡張し、ネットワークルーティングにより異なるノード機器間の相互接続を行う。

20

#### 【0025】

ブロックチェーンシステムにおいて1本、又は複数本のブロックチェーンを含むことができる。ブロックチェーンは、暗号学的方法を使用して関連付けることにより生じた一連のデータブロックであり、各々のデータブロック中には1バッチのネットワーク取引の情報が含まれ、その情報の有効性（偽造防止）を認証して次のブロックを生成することに用いられる。

#### 【0026】

ブロックチェーンシステムにおいて、ノード機器間はピアツーピア（Peer To Peer、P2P）ネットワークを構成することができ、P2Pプロトコルは伝送制御プロトコル（Transmission Control Protocol、TCP）において実行されている1つのアプリケーション層プロトコルである。ブロックチェーンシステムにおいて、いずれか1つのノード機器は以下の機能を備えることができる。

30

#### 【0027】

1）ルーティング：ノード機器が有した基本機能であり、ノード機器間の通信をサポートことに用いられる。

2）アプリケーション：ブロックチェーンにおいて配備されることに用いられ、実際のビジネスニーズに従って特定のビジネスを実現し、実現機能に関連するデータを記録して台帳データを形成する。台帳データにおいてデジタル署名が運ばれることでデータソースを表し、台帳データをブロックチェーンシステムにおけるその他のノード機器に送信して、その他のノード機器が台帳データソース、及び完全性の認証に成功すると、台帳データを一時ブロック中に追加する。ここで、アプリケーションにより実現されたビジネスは財布、共有台帳、及びスマートコントラクト等を含むことができる。

40

3）ブロックチェーン：前後の時間順序で相互につながった一連のブロックを含み、新しいブロックが一旦ブロックチェーンに加えられると、取り外されなくなり、ブロックにはブロックチェーンシステムにおけるノード機器がコミットした台帳データが記録されている。

#### 【0028】

いくつかの実施例において、各ブロック中には本ブロックが取引記録を記憶するハッシュ値（本ブロックのハッシュ値）、及び前のブロックのハッシュ値を含むことができ、各

50

ブロックはハッシュ値により接続されてブロックチェーンを形成し、また、ブロック中にはブロック生成時のタイムスタンプ等の情報をさらに含むことができる。

#### 【0029】

データベースシステムにおけるトランザクション並列制御の正確度は整合性、及び隔離性により記述することができるため、本願の実施例を紹介する前に、以下、先に整合性、及び隔離性に対して解釈して説明する。

#### 【0030】

1. 隔離性：トランザクション隔離レベルはあるデータ異常を回避できるか否かにより定義され、関する可能性があるデータ異常は以下を含む。A、ダーティリード：1つのトランザクションがもう1つのトランザクションからコミットされていないデータ項目を読み取ったことを指す。B、ノンリピータブルリード：1つのトランザクションが同一データ項目を2回繰り返し読み取ったが異なる結果を得ることを指す。C、ファントムリード：トランザクションが操作過程において2回述語問い合わせ（範囲問い合わせ）を行い、2回目の問い合わせ結果が1回目の問い合わせ結果において出現していないデータ項目を含むか、又は1回目の問い合わせ結果において出現したデータ項目が欠けることを指す。

#### 【0031】

上記ABCの3種類のデータ異常を解決することを可能にすることに基づいて、データベース国際標準国家標準学会（A-National Standards Institute、ANSI）SQLは4種類の隔離レベルを提案し、上記ABCの3種類の知られているデータ異常を区別し、あるいくつかのデータ異常が存在することを許可する場合にトランザクション処理効率を向上させる。

#### 【0032】

また、さらに4種類の隔離レベルを含み、それぞれ以下を含む。a、リード・アンコミットテッド：上記ABCの3種類のデータ異常の発生を許可する。b、リード・コミットテッド：ダーティリードの発生を許可しない。c、リピータブルリード：ダーティリード、ノンリピータブルリードの発生を許可しない。d、直列化可能：上記ABCの3種類のデータ異常がいずれも発生してはならない。このことから容易に分かるように、この4種類の隔離レベルはいずれもダーティライト異常の発生を許可せず、ダーティライト異常とは、2つのコミットされていないトランザクションが同一データ項目を修正することを指す。ANSI SQLが標準を制定する時に知られているデータ異常の種類は多くなく、後続で新しいデータ異常が絶えず発見される。上記いくつかのデータ異常に加えて、データ異常はさらに、紛失更新異常、読み取り半順序異常、書き込み半順序異常、読み取り書き込み半順序異常、鋸歯波書き込み半順序異常、直列並列現象（Serial-Concurrent-Phenomenon）異常、交差現象（Cross-Phenomenon）異常、因果紛失異常、因果反転異常、古い読み取り異常、及び将来読み取り異常等を含む。

#### 【0033】

2. 整合性：データベース技術において、「整合性」という言葉は、2つの意味を表すことができ、第一としてはトランザクションの整合性を指し、第二としては分散型整合性を指す。

#### 【0034】

データベースのトランザクション整合性については、トランザクションの操作下で、データベースのデータ状態が1つの整合状態からもう1つの整合状態に変更される。上記「整合状態」とは、データベースシステムが予め定義したいくつかの規則を満たすデータ状態を指し、たとえば、これらの規則は制約、カスケード、トリガー、及び三者間の任意の組み合わせ（データの論理セマンティクスに属する）を含むことができ、書き込み半順序異常が違反するのは特定データ間の制約であり、ここでの制約はユーザーセマンティクスにより限定されたデータの整合性に属する。

#### 【0035】

データベースのトランザクション処理技術における並列アクセス制御技術は、トランザ

クシオン並列操作がデータ項目上にデータ異常をもたらし得るか否か、及び如何にデータ異常を除去するかという問題を発見して解決することに用いられる。ANSI SQLは4種類のデータ異常、及び隔離レベルを提案し、各種の並列アクセス制御アルゴリズムをも発展させている。たとえば、ロッキングに基づく並列アクセス制御技術、タイムスタンプに基づく並列アクセス制御技術、マルチバージョン並列制御(Multi-Version Concurrency Control、MVCC)に基づく並列アクセス制御技術、楽観的並列制御(Optimistic Concurrency Control、OCC)に基づく並列アクセス制御技術等である。

#### 【0036】

データベースの分散型整合性は、共有データオブジェクトの操作整合性とも呼称され、データベースシステム全体に対するシステムレベルの意味である。データがデータベースシステムにおいて整合を保持することを保証するには、さらにデータベースシステムが2つの特性に適合する必要があることを指し、1つは直列化可能性(Serializability)であり、もう1つはリカバリ可能性(Recoverability)である。直列化可能性は、つまり、上記隔離性における直列化可能隔離レベルであり、直列化可能性は並列操作の正確性を保証し、リカバリ可能性とは、コミットされたトランザクションがロールバックされたトランザクションが書き込んだデータを読み取っていないことを指す(ダーティリード異常が発生しないことを指す)。リカバリ可能性はトランザクションがロールバックされた後にデータが前の整合状態に戻ることを保証し、ロールバックされたトランザクションはデータの整合性に影響をもたらさず、データベースシステムの整合性状態はリカバリ可能である。

#### 【0037】

上記概念に基づいて明らかなように、整合性と隔離性はいずれもデータ異常と密接に関連し、従って、データベースシステムにおいてデータ異常を識別し且つ回避する必要がある。ここでは、関連するデータベース異常は、ダーティリードデータ異常、ダーティライトデータ異常、ノンリピータブルリードデータ異常、ファントムリード異常、紛失更新データ異常、読み取り半順序データ異常、書き込み半順序データ異常、直列並列現象データ異常、及び交差現象データ異常等を含むことができる。

#### 【0038】

一般的には、データ異常を識別する時に直列化可能原理に依存し、通常、2種類のデータ異常識別戦略を採用することができる。第一としてはロッキング技術を利用し、ロックの排他メカニズムを介してデータ異常の発生を回避する。たとえば、DB2、Informix、及びMySQL等のデータベースシステムの直列化可能隔離レベルはいずれもロッキング技術を使用する。第二としては、依存グラフ技術を利用し、並列トランザクションにより構成された依存グラフにおいて閉路が存在するか否かを確認することにより、もし閉路が存在すれば、閉路の存在を打破し、それにより潜在的なデータ異常を除去する必要がある。

#### 【0039】

しかしながら、ロッキング技術はデータベースシステムの並列性を深刻に制限し、トランザクション処理効率の低下を引き起こす。また、依存グラフ技術は並列トランザクションの間に各トランザクションをトラバースルすることで閉路の存在を識別するため、トランザクション処理効率が比較的低いことを引き起こし、閉路が存在するか否かを識別するために、派生された各種のアルゴリズムの助けを借りる必要があり、且つ複雑なデータ構造で並列トランザクションと操作されるデータ項目(すなわち変数バージョン)との間の関連関係を表す必要があり、データベースシステムの複雑さが増大する。

#### 【0040】

また、トランザクション処理を行うために、さらに依存グラフ、及び競合直列化可能技術(該技術は直列化可能原理に基づく)に基づく簡略化された実現方式を採用することができる。たとえば、データベース(PostgreSQL)のスナップショット隔離(Snapshot Isolation、SSI)技術であり、SSI技術では依存グラフ

において閉路が存在することをもたらし可能性があり得る必要不十分条件を予め識別することを可能にする。1つのトランザクションに1本の読み取り書き込み競合入辺、及び読み取り書き込み競合出辺が存在し、もしいずれか1つのトランザクションが上記条件を満たすと、データ異常が存在すると考えられる。

#### 【0041】

また、データ異常を識別する上記方法には以下の問題がさらに存在する。

#### 【0042】

第1、認識が分断されて統一されていない。ロッキング技術、依存グラフ技術、及びSSI技術は、いずれも各種のデータ異常の識別方式を分断し、それによりデータ異常を識別する時に各種のデータ異常に整合するか否かを1つずつ認識しなければならない。つまり、統一された観点でこれらの異なる種類のデータ異常を識別、認識できず、データ異常に対する認識の構想が複雑で且つ不明瞭で統一されず、複雑度が比較的高いことを引き起こす。

10

#### 【0043】

第2、認識は限界があり全面的ではない。ロッキング技術、依存グラフ技術、及びSSI技術はいずれも未知の新しいデータ異常があるか否かを網羅することができない。データ異常が存在するときに、データベースシステム全体に損失をもたらすようになる。たとえば、データベースシステムが非直列化可能隔離レベル下にあると、新しいデータ異常があれば、システムにおいてデータが整合しないというエラーが出現することを引き起こすようになる。

20

#### 【0044】

第3、識別原理は正確ではない。直列化可能は、データに異常が存在することの十分不必要条件であり、このようにすると、データベースシステムの実現において新しいデータ異常の出現を回避することができるが、データ異常が存在しないいくつかの場合もデータ異常とする可能性があり、データ異常に対して誤判断を行うことに相当する。たとえば、トランザクション1は変数バージョンX1を書き込み、且つ変数バージョンY1を読み取り、トランザクション2は変数バージョンX1を読み取り、且つ変数バージョンY1を書き込み、上記一連のデータベース操作は「W1(X1) R2(X1) W2(Y1) R1(Y1) C2C1」として記すことができ、この一連の操作は直列化不可能であるが、データ異常が存在しない。

30

#### 【0045】

これに鑑み、本願の実施例はトランザクション処理方法を提供し、トランザクション処理過程においてデータ異常を識別し、且つデータ異常を回避する方式を提供しており、一種の汎用形式の競合閉路グラフを提供してデータ異常を定義し、データ異常の定義の正確性を確保する前提において、データ異常の定義を形象化し、且つ、データ異常に対して細分化分類を行い、データベースシステムのトランザクション処理効率を向上させることを可能にすることについて、以下、詳しく説明する。

#### 【0046】

図1は、本願の実施例により提供されたトランザクション処理方法の実施環境の模式図である。図1に参照されるように、本願の実施例はデータベースシステムにおける分散型データベースシステムに適用することができ、該システムにおいて、ゲートウェイサーバ101（例示的には3つのゲートウェイサーバを示している）、グローバルタイムスタンプ生成クラスター102、分散型記憶クラスター103、及び分散型協調システム104（たとえばZookeeper）を含むことができる。分散型記憶クラスター103においてノード機器1～m（mが1以上の正整数である）を含むことができ、各ノード機器はデータノード機器であってもよく、さらに協調ノード機器であってもよい。

40

#### 【0047】

ここで、ゲートウェイサーバ101は外部の読み取り書き込み要求を受信し、且つ読み取り書き込み要求に対応する読み取り書き込みトランザクションを分散型記憶クラスター103に配信することに用いられる。たとえば、ユーザーは端末上のアプリケーションク

50

クライアント端末にログインした後に、アプリケーションクライアント端末をトリガーして読み取り書き込み要求を生成し、分散型データベースシステムから提供されたアプリケーションプログラムプログラミングインタフェース (Application Programming Interface、API) を呼び出して該読み取り書き込み要求をゲートウェイサーバ101に送信する。ここで、APIは任意の一種のリレーショナルデータベースシステムにより提供されたAPIであってもよく、たとえば、該APIはMySQL APIであってもよい。

【0048】

いくつかの実施例において、該ゲートウェイサーバ101は分散型記憶クラスター103におけるいずれか1つのデータノード機器、又はいずれか1つの協調ノード機器と同一物理マシンであってもよく、すなわち、あるデータノード機器、又は協調ノード機器にゲートウェイサーバ101の機能を兼ね備える。

【0049】

グローバルタイムスタンプ生成クラスター102は、グローバルトランザクションのグローバルコミットタイムスタンプ (Global Timestamp、Gts) を生成することに用いられる。該グローバルトランザクションは分散型トランザクションとも呼称され、複数のデータノード機器に関するトランザクションを指し、たとえばグローバル読み取りトランザクションは複数のデータノード機器上に記憶されるデータの読み取りにすることができ、また、たとえば、グローバル書き込みトランザクションは複数のデータノード機器上のデータの書き込みに関することができる。グローバルタイムスタンプ生成クラスター102は論理的に1つの単一点と見なすことができ、さらに1マスター3スレーブ (それぞれ1つのマスター機器、及び該マスター機器に対応する3つのスレーブ機器) のアーキテクチャにより、より高い可用性を有するサービスを提供してもよく、クラスター形式を採用して該グローバルコミットタイムスタンプの生成を実現し、1マスター3スレーブアーキテクチャにおける1つの機器が故障するときに、1マスター3スレーブアーキテクチャにおけるその他の機器により該グローバルコミットタイムスタンプの生成を継続的に実現することができ、それにより単一点故障を防止することができ、単一点ボトルネック問題も回避する。

【0050】

本願の実施例において、グローバルコミットタイムスタンプとは、分散型データベースシステムにおいてグローバルに一意で且つ単調に逡増した1つのタイムスタンプ識別子であり、各トランザクションのグローバルコミットの順序をマークすることに用いることができ、これにより、トランザクション間が真の時間上での前後関係 (トランザクションの全順序関係) を反映する。グローバルコミットタイムスタンプは物理クロック、論理クロック、ハイブリッド物理クロック、又はハイブリッド論理クロック (Hybrid Logical Clock、HLC) のうちの少なくとも一項を採用することができ、本願の実施例はグローバルコミットタイムスタンプのタイプについて限定しない。

【0051】

例示的には、グローバルコミットタイムスタンプはハイブリッド物理クロック方式を採用して生成することができ、グローバルコミットタイムスタンプは8バイトで構成することができ、各バイトは8ビットを含み、合計64ビットである。ここで、前の44ビットは物理タイムスタンプの取る値 (すなわちUnixタイムスタンプであり、ミリ秒まで正確) であってもよく、このように合計 $2^{44}$ 個の符号なし整数を表すことができ、従って、理論的には合計で約557.8 ([数1]) 年の物理タイムスタンプを表すことができる。

【0052】

【数1】

10

20

30

40

50

$$\frac{2^{44}}{1000 * 60 * 60 * 24 * 365} = 557.8$$

#### 【0053】

ここで、後の20ビットはあるミリ秒内での単調な増減カウントとすることができ、このように、1ミリ秒あたりに $2^{20}$ 個（約100万個）のカウントを有する。上記データ構造に基づいて、もしスタンドアロン（いずれか1つのデータノード機器）のトランザクションスループットが10w/sであれば、理論的には1万個のノード機器を含む分散型記憶クラスター103をサポートすることができる。同時に、グローバルコミットタイムスタンプの数量はシステムが理論的にサポートできるトランザクション総数を表し、上記データ構造に基づいて、理論的にはシステムは $(2^{44} - 1) * 2^{20}$ 個のトランザクションをサポートすることができる。これは一種のグローバルコミットタイムスタンプの定義方法についての例示的な説明である。ビジネスニーズの相違に従って、グローバルコミットタイムスタンプのビット数を拡張することができ、それによりより多くのノード数、トランザクション処理数へのサポートを満たすため、本願の実施例はグローバルコミットタイムスタンプの定義方法について限定しない。

#### 【0054】

いくつかの実施例において、該グローバルタイムスタンプ生成クラスター102は物理的に独立していてもよいし、分散型協調システム104と一体に合併されてもよい。

#### 【0055】

ここで、分散型記憶クラスター103は、データノード機器、及び協調ノード機器を含むことができる。各協調ノード機器は少なくとも1つのデータノード機器に対応することができ、データノード機器と協調ノード機器との区分はトランザクションの異なりに基づいて区分される。あるグローバルトランザクションを例とすると、グローバルトランザクションの開始ノードは協調ノード機器と呼称することができ、グローバルトランザクションに関するその他のノード機器はデータノード機器と呼称され、データノード機器、又は協調ノード機器の数量は1つ、又は複数であってもよく、本願の実施例は分散型記憶クラスター103におけるデータノード機器、又は協調ノード機器の数量について限定しない。本願の実施例が提供した分散型データベースシステムにおいては、グローバルトランザクションマネージャーを欠いているため、該システムにおいて、組織分散型トランザクション規範（eXtended Architecture、XA）／二相コミット（Two-Phase Commit、2PC）技術を採用してノード間のトランザクション（グローバルトランザクション）をサポートすることができ、ノード間の書き込み操作時のデータの原子性、及び整合性を保証することができる。このときに、協調ノード機器は2PCアルゴリズムにおける協調者に相当し、該協調ノード機器と対応する各データノード機器は2PCアルゴリズムにおける関与者に相当する。

#### 【0056】

本願の実施例において、各データノード機器、又は協調ノード機器はスタンドアロン機器であってもよく、マスタースタンバイ構造（すなわち、シングルマスターマルチスタンバイクラスター）を採用してもよい。図1に示すように、ノード機器（データノード機器、又は協調ノード機器）は、1マスター2スタンバイクラスターであることを例として例示すると、各ノード機器中には1つのマスター機器と2つのスタンバイ機器とを含み、各マスター機器、又はスタンバイ機器にはいずれもエージェント（Agent）機器が対応して配置されている。エージェント機器はマスター機器、又はスタンバイ機器とは異なる物理機器であってもよく、もちろん、エージェント機器はさらにマスター機器、又はスタンバイ機器上の1つのエイジェントモジュールとすることもできる。ノード機器1を例とすると、ノード機器1は1つのマスターデータベース、及びエージェント機器（マスターDatabase + Agent、マスターDB + Agentと略称）を含み、また、2

つのスタンバイデータベース、及びエージェント機器（スタンバイ Database + Agent、スタンバイ DB + Agent と略称する）をさらに含む。

【0057】

例示的には、各ノード機器と対応するマスター機器、又はスタンバイ機器のデータベースインスタンス集合は、1つの集合（SET）と呼称され、たとえば、あるノード機器がスタンドアロン機器であると仮定すると、該ノード機器のSETは該スタンドアロン機器のデータベースインスタンスだけである。あるノード機器が1マスター2スタンバイ機器クラスターであると仮定すると、該ノード機器のSETはマスター機器データベースインスタンス、及び2つのスタンバイ機器データベースインスタンスの集合である。このときに、クラウドデータベースの強同期技術に基づいてマスター機器のデータとスタンバイ機器のレプリカデータとの間の整合性を保証することができ、ここでは、各SETは線形拡張を行うことができ、それによりビッグデータシーン下でのビジネス処理ニーズに対処する。いくつかの金融ビジネスシーン下では、グローバルトランザクションとは通常、SET間の振替を指す。

【0058】

分散型協調システム104はゲートウェイサーバ101、グローバルタイムスタンプ生成クラスター102、又は分散型記憶クラスター103のうちの少なくとも一項を管理することに用いることができる。本願の実施例において、端末上のスケジューラ（Scheduler）により該分散型協調システム104にアクセスし、それによりフロントエンドのスケジューラに基づいてバックエンドの分散型協調システム104を制御して、各々のクラスター、又はサーバへの管理を実現することができる。たとえば、スケジューラによりZooKeeperを制御してある1つのノード機器を分散型記憶クラスター103中から削除し、すなわち、ある1つのノード機器を無効化することができる。

【0059】

上記図1は、一種の軽量レベルのグローバルトランザクション処理のアーキテクチャ図を提供しており、一種の分散型データベースシステムである。分散型データベースシステム全体は1つの論理上のデータテーブルを共同で維持するものとみなすことができ、このデータテーブルにおいて記憶されたデータは主キーにより分散型記憶クラスター103における各々のノード機器中に散布され、各ノード機器上に記憶されたデータはその他のノード機器から独立したものであり、それによりノード機器の論理データテーブルに対する水平分割を実現する。上記システムにおいて各々のデータベースにおける各々のデータテーブルを水平分割した後に分散的に記憶することを可能にし、従って、このようなシステムも「データベース分割テーブル分割」を有するアーキテクチャとも呼称することができる。

【0060】

上記分散型データベースシステムにおいては、既にXA/2PC技術に基づいて書き込み操作時のデータの原子性、及び整合性を実現している。たとえば、図1において示された分散テーブル分割アーキテクチャは、1つの局所トランザクションマネージャーであり、グローバルなものではなく、従って、読み取り操作時のデータの整合性問題が未だ存在する。一方、本願の実施例により提供されたトランザクション処理方法は、分散型トランザクション処理の実現を可能にし、且つ軽量で、非集中化された分散型トランザクション処理メカニズムの構築により、分散型データベースシステムに水平拡張等の能力を提供することを可能にし、分散型データベースシステムの応用範囲を向上させ、さらに書き込み操作時のデータの原子性、及び整合性、及び読み取り操作時のデータの整合性問題を解決することを可能にし、それによりトランザクション処理効率を向上させる。

【0061】

本願の実施例により提供されたトランザクション処理方法は、上記したデータベース分割テーブル分割アーキテクチャを採用した分散型データベースシステムに応用することができる。たとえば、該分散型データベースシステムは分散型トランザクション型データベースシステムであり、もちろん、分散型リレーショナルデータベースシステムであっても

よく、また、本願の実施例により提供されたトランザクション処理方法はいくつかのスタンドアロンデータベースシステムにも応用することができる。本願の実施例により提供されたトランザクション処理方法は分散型データベースシステムに応用されるときに、分散型トランザクション処理を実現することを可能にし、且つ柔軟な隔離レベルにより、トランザクション処理効率を向上させることを可能にする。

#### 【0062】

いくつかの実施例において、上記ゲートウェイサーバ101、グローバルタイムスタンプ生成クラスター102、分散型記憶クラスター103、及び分散型協調システム104から構成された分散型データベースシステムは、ユーザー端末にデータサービスを提供する一種のサーバと見なすことができる。該サーバは独立した物理サーバであってもよく、複数の物理サーバで構成されたサーバクラスター、又は分散型システムであってもよく、さらにクラウドサービス、クラウドデータベース、クラウドコンピューティング、クラウド関数、クラウド記憶、ネットワークサービス、クラウド通信、ミドルウェアサービス、ドメイン名サービス、安全サービス、内容配信ネットワーク(Content Delivery Network、CDN)、ビッグデータ、及び人工知能プラットフォーム等の基本クラウドコンピューティングサービスを提供するクラウドサーバであってもよい。ここでは、上記ユーザー端末は、スマートフォン、タブレットコンピュータ、ノートパソコン、デスクトップコンピュータ、スマートスピーカー、及びスマートウォッチ等であってもよいが、これらに制限されない。端末、及びサーバは有線、又は無線通信方式により直接、又は間接的に接続することができ、本願はここで制限しない。

#### 【0063】

以下、データベースシステムにおいて関したいくつかの基本用語、及び符号表現を紹介する。

#### 【0064】

トランザクション：データベース管理システムが操作を実行する過程における1つの論理単位であり、1つの限られたデータベース操作シーケンスにより構成され、データベースシステム操作の最小実行単位である。

#### 【0065】

変数：トランザクションはデータベース関係システムにおける1つのデータ単位であり、変数はデータベース操作の作用者（又は操作されるデータオブジェクトともいう）であり、いくつかの実施例において、変数はデータ項目とも呼称される。ここで、1つの変数は1つのタプル(Tuple)、又は記録(Record)であってもよく、1つのページ(Page)、又は1つのテーブル(Table)オブジェクト等であってもよい。1つの変数は複数の変数バージョン（「バージョン」と略称）を含むことができ、トランザクションが変数を更新するたびに、新しい変数バージョンを追加することになり、変数の各々の変数バージョンは自然数をバージョン番号識別子とすることができ、バージョン番号が大きければ大きいほど、変数バージョンが新しくなることを表す。

#### 【0066】

操作：1つのデータベース操作は操作タイプ、トランザクション、及び変数バージョンの3つの部分により構成される。ここで、操作タイプは読み取り(Read、R)、及び書き込み(Write、W)という2種類を含むことができる。たとえば、トランザクション $T_i$ は変数 $x$ を読み取り、変数 $x$ の新しいバージョン $x_m$ を生成し、上記読み取り操作は $R_i[x_m]$ と示すことができ、また、たとえば、トランザクション $T_j$ は変数 $y$ を更新し、変数 $y$ の新しいバージョン $y_n$ を生成し、上記書き込み操作は $W_j[y_n]$ と記すことができる。ここで、 $i$ 及び $j$ は自然数であり、それぞれ対応する操作が所属するトランザクション番号であり、 $m$ 及び $n$ は自然数であり、それぞれ対応する変数のバージョン番号である。また、 $R$ と $W$ を区別する必要がないときに、 $O$ で1つの操作を代表することもできる。

#### 【0067】

直列化可能：直列化可能は1つのスケジュールであり、複数のトランザクション間の実

行方式として理解することもできる。複数のトランザクション間の実行には前後順序があり、もしトランザクション間に共同で操作される変数がなければ、トランザクション間の実行順序の前後を置換することは構わない。しかし、もしトランザクション間に共同で操作される変数が存在すれば、トランザクション間の実行の前後の順序は区別する必要がある。共同で操作される変数が存在する並列して実行される複数のトランザクションに対しては、もしその実行結果がある「直列化スケジュール」と「等価」であれば、このスケジュールは「直列化可能スケジュール」である。「直列化可能スケジュール」を満たすと、直列化可能属性を有する。従って、直列化可能は複数のトランザクションの並列時の実行順序がデータの整合性に影響しないことを保証する。

【0068】

データベースシステムにおける各トランザクションは2つのトランザクションデータ集合を持ち、それぞれトランザクションの書き込みセット、及びトランザクションの読み取りセットであり、その意味は、書き込みセット  $DSW(T)$  はトランザクション  $T$  により書込まれた新しいデータバージョンを記憶することに用いられ、読み取りセット  $DSR(T)$  はトランザクション  $T$  により読み取られたデータバージョンを記憶することに用いられる、ということである。

【0069】

本願の実施例が提供したトランザクション処理方法は、データ状態整合性モデル（「データ状態モデル」と略称することもできる）を基礎として実行され、本願の実施例は競合直列化可能理論に基づいて、データ異常を統一し、且つ汎用形式の競合閉路グラフを提供しており、トランザクション間の関連関係、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられる。以下、本願に関する基礎的な概念を紹介する。

【0070】

1、スケジュール：1つのスケジュールとは1組のトランザクションの複数の操作からなるシーケンスを指し、 $s$  と記される。1つのスケジュール  $s$  におけるすべてのトランザクションは  $TS(s)$  と記され、1つのスケジュール  $s$  におけるすべての操作は  $O(s)$  と記され、1つのスケジュール  $s$  におけるバージョン番号識別子変数付きのすべての集合は  $VS(s)$  と記される。たとえば、スケジュール  $s = R_1[x_0] R_2[y_0] W_3[x_1] W_4[y_1] R_2[x_1] R_1[y_1]$  は、該スケジュール  $s$  の操作シーケンスが4つのトランザクション、すなわち  $TS\{T_1, T_2, T_3, T_4\}$  から由来することを表し、2つの変数  $x$  及び  $y$  に関し、且つ変数  $x$  及び  $y$  はそれぞれ2つのバージョンを有する。

【0071】

例示的には、もし1つのスケジュール  $s$  において、任意の2つのトランザクションの操作のそれぞれはトランザクションを単位として隣接し、すなわち、1つのトランザクションのすべての操作を実行した後に次のトランザクションをさらに実行すると、該スケジュール  $s$  を直列化スケジュールと呼ぶ。たとえば、1つの直列化スケジュール  $s$  は  $s = R_1[x_0] R_1[y_0] R_2[y_0] R_2[x_0] W_3[x_1] W_4[y_1]$  で表される。

【0072】

2、競合 (Conflict)：競合は競合操作とも呼称され、2つの操作が以下の3つの条件を満たすと、この2つの操作を競合操作と呼ぶ。ここで、3つの条件は、1) 2つの操作は異なるトランザクションに属する、2) 少なくとも1つの操作は書き込み操作である、及び3) 2つの操作が操作したオブジェクトが同一オブジェクトである（同一オブジェクトを読み取る、又は同一オブジェクトを書き込む）、であり、本願の実施例において、オブジェクトは変数とも呼称される。

【0073】

例示的には、仮に  $O_i[x_m] O_j[x_n]$  ( $i \neq j, m < n$ ) はスケジュール  $s$  における2つのトランザクションの操作であり、それぞれトランザクション  $T_i$  及び  $T_j$  が同一変数  $x$  上に作用する2つの操作であるとする。もし  $O_i[x_m] O_j[x_n]$  が  $R_i[x_m] W_j[x_{m+1}]$ 、 $W_i[x_m] R_j[x_m]$ 、及び  $W_i[x_m] W_j[x_{m+1}]$  の3種

10

20

30

40

50

類の競合のうちのいずれかに属すれば、 $O_i[x_m] O_j[x_n]$  を競合と呼び、競合は  $O_i$  と  $O_j$  との間に半順序関係を構成し、スケジュール  $s$  におけるすべての競合を含む集合を  $conf(s)$  と記す。本願の実施例において、上記 3 種類の競合は順にそれぞれ  $RW$ 、 $WR$ 、及び  $WW$  と略記することができる。

【0074】

3、競合等価 (Conflict Equivalence) : 異なるスケジュール  $s$  及び  $s'$  に対して、もし以下の 2 つの条件を満たせば、 $s$  及び  $s'$  を競合等価と呼ぶ。ここで、2 つの条件は、1) スケジュール  $s$  及び  $s'$  が同様のトランザクション集合を含み、すなわち、各トランザクションにおける操作の順序が固定され、異なるスケジュール下で変化できない、及び 2) スケジュール  $s$  及び  $s'$  は同様の競合集合を含む、である。

【0075】

例示的には、異なるスケジュール  $s$  及び  $s'$  に対しては、もし  $O(s) = O(s')$ 、且つ  $conf(s) = conf(s')$  であれば、 $s$  及び  $s'$  を競合等価と呼び、 $s \sim s'$  と記す。

【0076】

たとえば、以下、2 つの競合等価のスケジュールである。

【0077】

$s = R_1[x_0] R_1[y_0] W_2[x_1] W_1[y_1] R_2[z_0] W_1[x_2] W_2[y_2]$ 、  
 $s' = R_1[y_0] R_1[x_0] W_1[y_1] W_2[x_1] W_1[x_2] R_2[z_0] W_2[y_2]$ 。

【0078】

4、競合直列化可能 (Conflict Serializability) : もし 1 つのスケジュール  $s$  に 1 つの直列化スケジュール  $s'$  が存在し、この 2 つのスケジュールを競合等価にすると (すなわち、 $s \sim s'$  を満たす)、 $s$  を競合直列化可能スケジュールと呼ぶ。

【0079】

5、競合閉路グラフ (Conflict Cycle Graph、CCG) : 例示的には、1 つのスケジュール  $s$  において、複数のトランザクション  $t \in TS(s)$  が存在し、もし複数のトランザクション  $t$  で構成されたグラフは以下の 3 つの条件を満たせば、該閉路グラフを競合閉路グラフと呼び、ここで、 $CCG(s) = (V, E)$  である。以下、3 つの条件を説明する。

【0080】

1)  $V = VS(s)$  は、CCG グラフの頂点セットとして表され、該頂点セットにおける要素は頂点と呼称され、1 つの頂点は 1 つのトランザクションに対応し、ここで、1 つの頂点  $T_i$  は  $T_i \in TS(s)$  を満たす。

【0081】

2)  $E$  は  $V \times V$  のサブセットであり、辺セットとも呼称することができ、該辺セットにおける要素は有向辺と呼称され、 $\langle T_i, T_j \rangle$  と記され、辺と略称することができ、ここで、 $\langle T_i, T_j \rangle \in E \Leftrightarrow i \neq j \wedge (\exists O_i \in T_i) (\exists O_j \in T_j) (O_i, O_j) \in conf(s)$  であり、従って、有向辺は上記「競合」と類似する形式と記すことができる。たとえば、有向辺を  $W_i[x_m] W_j[x_{m+1}]$  と記し、本願の実施例はこれについて限定しない。

【0082】

3)  $\langle T_i, T_j \rangle \in E, \langle T_j, T_k \rangle \in E, \dots \langle T_p, T_i \rangle \in E, i \neq j \neq k \neq \dots \neq p$  であると、CCG グラフは有向閉路で構成された 1 つのグラフである。

【0083】

たとえば、図 2 に参照されるように、図 2 は本願の実施例により提供された競合閉路グラフの模式図である。図 2 に示すように、図における  $T_1$  及び  $T_2$  はそれぞれ競合閉路グラフ 2-1 の頂点であり、且つ  $T_1$  と  $T_2$  との間は 2 本の辺により連結され、競合閉路グ

10

20

30

40

50

ラフを構成する。ここで、2本の辺はそれぞれ  $R_1 [x_0] W_2 [x_1]$  及び  $W_2 [x_1] R_1 [x_1]$  で表される。

【0084】

本願の実施例において、競合閉路グラフに基づいて、以下の3種類の推論を得て、以下、それぞれ説明する。

【0085】

1) 1つの競合閉路グラフは少なくとも2本の辺を有する。

【0086】

例示的には、もし競合閉路グラフに1本の辺のみがあれば、競合閉路グラフにおいて2つのトランザクションのみが存在し、且つこの2つのトランザクションは直列化トランザクションであり、従って、有向閉路を構成することがない。もし2本の辺が有向閉路を構成し、たとえば、競合閉路グラフにおける1つ目の辺は  $T_i$  を  $T_j$  より前に発生させ、競合閉路グラフにおける2つ目の辺は  $T_j$  を  $T_i$  より前に発生させる、すなわち等価な直列化可能スケジュールが存在しなければ、この2つのトランザクション  $T_i$  と  $T_j$  は競合する。

【0087】

2) 1つの競合閉路グラフは最大で無数の辺がある。

【0088】

例示的には、1つのスケジュール  $s$  に対応する  $TS(s)$  において無数のトランザクションが存在することができ、相応には、1つの競合閉路グラフにおいて無数の頂点が存在することができ、すなわち、無数の辺が存在することができる。

【0089】

3) もし1つの競合閉路グラフはいずれか1本の辺を除去すれば、閉路を構成しない。

【0090】

例示的には、上記競合閉路グラフの3番目の条件に従ってわかるように、競合閉路グラフで構成された有向閉路において、 $i \neq j \neq k \neq \dots \neq p$  が存在し、すなわち、任意の2本の辺は異なり、従って、1つの競合閉路グラフに対して、いずれか1本の辺を除去すれば、閉路を構成しない。

【0091】

6、競合整合性状態グラフ (Conflict Consistency State Graph、CCSG) : 例示的には、1つのスケジュール  $s$  において、複数のトランザクション  $t \in TS(s)$  が存在し、もし複数のトランザクション  $t$  で構成されたグラフは以下の4つの条件を満たせば、該グラフを競合整合性状態グラフと呼び、ここで、 $CCSG(s) = (V, E)$  である。以下、4つの条件をそれぞれ説明する。

【0092】

1) CCSGグラフにおけるスケジュール  $s$  の定義は上記CCGグラフにおけるスケジュール  $s$  の定義を満たし、すなわち  $CCG(s)$  と  $CCSG(s)$  との間は等価変換することができる。

【0093】

2)  $CCSG(s)$  に対して、 $V = VS(s)$  はCCSGグラフの頂点セットと呼称され、その要素は頂点と呼称される。1つの頂点  $v_i$  は  $v_i \in VS(s)$  を満たし、該頂点は2種類のタイプのデータにより構成され、第一としてはトランザクション識別子 (Identification、ID) であり、第二として1つのリスト (List) であり、該リストは変数の異なるバージョンを保存することに用いられる。

【0094】

3)  $E$  は  $V \times V$  のサブセットであり、辺セットとも呼称することができ、該辺セットにおける要素は有向辺と呼称され、CCSGグラフにおいて、辺は以下の2種類に分けられ、以下の2種類の辺に対して、 $\langle v_i, v_j \rangle$  として統一して表現される。このような辺はCCGグラフの辺の等価変形から由来しており、従って、競合操作の変形に属する。以下、2種類の辺をそれぞれ説明する。

【0095】

10

20

30

40

50

第1種の辺は整合性状態辺であり、 $\langle x_m, y_n \rangle$ と記され、トランザクション辺  $E_T$  と略称される。該タイプの辺は同一トランザクションの操作を含み、従って、該トランザクション内のデータ状態は整合性要求を満たす。ここで、 $x = y$  且つ  $m \neq n$ 、又は  $x \neq y$  である。CCSGグラフはCCGグラフと等価であるため、整合性状態辺は  $R_i[x_m] R_i[y_n]$  として表されてもよく、 $E_T$  は該辺のラベルとすることができる。

【0096】

第2種の辺は状態遷移辺であり、 $\langle x_i, x_{i+1} \rangle$ と記され、状態辺  $E_S$  と略称される。該タイプの辺はある書き込みトランザクションがある変数を書き込んでデータ状態の遷移を発生させることにより構成され、従って、変数のバージョンは変化する。CCSGグラフはCCGグラフと等価であるため、整合性状態辺は  $W_j[x_{m+1}]$  として表されてもよい。

【0097】

4)  $\exists V_1 \in V, V_2 \in V \dots, V_k \in V$  であり、 $\exists E_1 \in E, E_2 \in E \dots, E_k \in E$  でもあり、それによりCCSGグラフは1つの有向閉路により構成されるグラフである。すなわち、 $V_1 \rightarrow E_1 \rightarrow V_2 \rightarrow E_2 \dots V_k \rightarrow E_k \rightarrow V_1$  は有向閉路で構成された1つのグラフである。

【0098】

7、CCGグラフからCCSGグラフへの変換の等価変換規則：ここで、CCGグラフからCCSGグラフへの等価変換規則は以下の4項を含み、以下、それぞれ説明する。

【0099】

第1項は辺分割である。ここで、CCGグラフのいずれか1本の辺は、2つの操作すなわち1つの競合により構成され、それは2つの単独した辺に分割することができる。例示的には、辺の分割方式は以下の3種類を含み、各種において2つのサブ辺が分割され、以下、3種類の分割方式を説明する。

【0100】

1) たとえば、辺は  $R_1[x_0] W_2[x_1]$  で表され、該辺を分割して、 $R_1[x_0]$  及び  $W_2[x_1]$  を得て、同時に変数  $x$  の2つのバージョン  $x_0$  及び  $x_1$  を得る。

【0101】

2) たとえば、辺は  $W_2[y_1] R_3[y_1]$  で表され、該辺を分割して、 $W_2[y_1]$  及び  $R_3[y_1]$  を得て、同時に変数  $y$  の1つのバージョン  $y_1$  を得る。

【0102】

3) たとえば、辺は  $W_3[z_1] W_1[z_2]$  で表され、該辺を分割して、 $W_3[z_1]$  及び  $W_1[z_2]$  を得て、同時に変数  $z$  の2つのバージョン  $z_1$  及び  $z_2$  を得る。

【0103】

第2項は頂点変換である。ここで、変数のバージョンをCCSGグラフの頂点として、すなわち、前のステップのバージョンを頂点に変換する。ここでは、上記第1項の例示で得られた頂点を例とし、CCSGグラフの頂点は  $x_0, x_1, y_1, z_1$  及び  $z_2$  を含む。同一変数の複数のバージョンは縦方向に並んでおり、同一トランザクションにより作用される異なる変数のバージョン間は横方向に並んでおり、ここで、縦方向とは、スクリーンに表示される情報に対応する水平方向を指してもよく、さらにスクリーンに表示される情報に対応する水平方向に垂直な方向を指してもよく、本願の実施例はこれについて限定しないが、同一変数の複数のバージョンの並び方向は、同一トランザクションにより作用される異なる変数のバージョン間の並び方向に対向する。説明する必要がある点としては、ここでの頂点の並び方式は特定の要件ではなく、いくつかの実施例において、さらにその他の並び方式で並んでもよく、本願の実施例はこれについて限定しない。

【0104】

第3項は隣接するバージョンの同一変数に対応する頂点の間に辺を追加することである。ここで、上記第2項で得られた新しい頂点間の遷移関係をCCSGグラフの新しい辺とする。たとえば、上記トランザクション  $T_2$  の関連サブ辺は、 $W_2[x_1]$  及び  $W_2[y_1]$  を含み、この場合、 $x_1$  及び  $y_1$  を水平に並列させ、それにより  $x_0$  と  $x_1$  との間に1本

10

20

30

40

50

の  $W_2 [x_1]$  辺が存在する。

【0105】

第4項は同一トランザクションが作用した変数に対応する頂点の間に辺を追加することである。ここで、同一トランザクションの間に、CCSGグラフの整合性状態辺を構築する。たとえば、継続的に上記頂点を例として、 $x_1$ と $y_1$ との間に辺を追加し、その辺のラベルはトランザクション $T_2$ である。

【0106】

8、CCSGグラフからCCGグラフへの変換の等価変換規則：ここで、CCSGグラフからCCGグラフへの等価変換規則は主に以下の3項を含み、以下、それぞれ説明する。

【0107】

第1項は頂点の合併である。ここで、CCSGグラフに対して、同一トランザクションが操作した変数を1つの頂点に合併し、且つ該頂点を該トランザクションのトランザクション識別子として命名し、該頂点の表し方は、上記CCGグラフの頂点の表し方式と等価であり、ここでは再度繰り返し記述しない。

【0108】

第2項は隣接する頂点の間に辺を追加することであり、以下の2つの点を含み、第1点としては変数の状態遷移を残す。たとえば、辺 $R_1 [x_0] W_2 [x_1]$ から明らかなように、変数の状態遷移は $T_1$ から $T_2$ へ指し、従って、追加した新しい辺は $T_1 \rightarrow T_2$ である。第2点としては、もし書き込み操作が発生しなければ、元のCCGグラフにおいて発生した競合関係に応じて新しい辺を構築することである。説明する必要がある点としては、WR競合関係が存在する場合に応用され、WR競合関係が存在する場合に、対応する辺はWの所在するトランザクションからRの所在するトランザクションへ指す。

【0109】

第三としては得られたCCGグラフに閉路が存在し、且つその有向属性を無視することができ、すなわち、CCSGを変換することにより得られたCCGグラフは1つの無向有閉路グラフである。

【0110】

説明する必要がある点としては、本願の実施例において、CCSGグラフはすなわちデータ状態整合性モデルの基礎である。すなわち、変数のバージョンを頂点とし、トランザクションの読み取り書き込み操作を状態遷移の辺とするか、又はトランザクションの整合性状態辺を辺として、構成した図は、すなわちデータ状態整合性モデルである。該データ状態整合性モデルにおいて、もし閉路が存在すれば、データ異常が存在し、もし閉路が存在しなければ、データ異常が存在しない。

【0111】

9、データ異常 (Data Anomalies、DA)：1つのスケジュール $s$ において、もし1つのサブスケジュール $s'$ が存在し、すなわち、 $\exists s' \subseteq s$ であり、且つ $s'$ に競合閉路グラフが存在すれば、スケジュール $s$ にデータ異常が存在すると言え、 $s'$ はある1つの具体的なデータ異常である。

【0112】

また、上記説明された競合閉路グラフ、及びデータ異常に基づいて、以下の3種類の推論を得られ、以下、それぞれ説明する。

【0113】

1) データ異常は無数である。競合閉路グラフにおける2番目の推論に基づいて分かるように、競合閉路グラフ中には無数の頂点、及び無数の辺が存在することができ、従って、データ異常は無数である。

【0114】

2) 無数の細分類されたデータ異常が存在する。競合閉路グラフにおける頂点が異なると、データ異常が異なり、競合閉路グラフにおける辺が異なると、データ異常が異なり、競合閉路グラフにおける頂点、及び辺のうちの少なくとも一種の個数が異なると、データ異常は異なる。従って、無数の細分類されたデータ異常が存在する。

10

20

30

40

50

## 【0115】

3) 形式化表現式を採用して各細分類されたデータ異常を表現し、且つ該細分類されたデータ異常を命名する。

## 【0116】

理解すべき点としては、1つのデータ異常は競合閉路グラフが存在するスケジュールsに対応するサブセットであり、従って、1つの細粒度のデータ異常は競合閉路グラフを構成する該スケジュールsのサブセット部分の表現式で表すことができる。

## 【0117】

たとえば、紛失更新異常を例とすると、該紛失更新異常に対応する競合閉路グラフは図3に参照されており、図3は本願の実施例により提供された他の競合閉路グラフの模式図である。図3に示すように、該競合閉路グラフ3-1において、 $V = \{T_1, T_2\}$ 、 $E = \{ \langle R_1[x_0] W_2[x_1] \rangle, \langle W_2[x_1] W_1[x_2] \rangle \}$ であり、この場合、該紛失更新異常の形式化定義は  $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_i[x_{m+2}]$  である。

## 【0118】

また、たとえば、読み取り半順序異常を例とすると、該読み取り半順序異常に対応する競合閉路グラフは図4に参照されており、図4は本願の実施例により提供された別の競合閉路グラフの模式図である。図4に示すように、該競合閉路グラフ4-1において、 $V = \{T_1, T_2\}$ 、 $E = \{ \langle R_1[x_0] W_2[x_1] \rangle, \langle W_2[y_1] R_1[y_1] \rangle \}$ であり、この場合、該読み取り半順序異常の形式化定義は  $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_i[y_n] \cdots R_i[y_n]$  である。

## 【0119】

説明する必要がある点としては、ここでは、上記2種類のデータ異常を例として、対応する競合閉路グラフを組み合わせて、データ異常の形式化に対して例を挙げて説明しており、その他のデータ異常に対応する競合閉路グラフ、及びデータ異常の形式化定義については後続で紹介する。

## 【0120】

10、ロールバック率：ロッキング技術を利用して並列を抑制し且つデータ異常の発生を回避するときに、本来コミットすることができるトランザクションがロールバックされるようになり、つまり、偽ロールバックとなる。以下、本願の実施例に関するロールバックの情報をそれぞれ説明する。

## 【0121】

1) データ異常ロールバック：スケジュールsに対して、データ異常の出現により引き起こされたトランザクションロールバックは、データ異常ロールバックと呼称され、真ロールバック (True Rollback、TR) とも呼称される。

## 【0122】

2) アルゴリズムロールバック：スケジュールsに対して、データ異常が存在しないときに、並列アクセスアルゴリズムによりデータの不整合が発生する可能性があるとして決定することにより引き起こされるトランザクションロールバックは、アルゴリズムロールバックと呼称され、偽ロールバック (False Rollback、FR) とも呼称される。

## 【0123】

3) 真ロールバック率：P個のスケジュールにおいて、もしU個のスケジュールのうちの各スケジュールはいずれも真ロールバックが存在すると仮定すれば、 $U/P$ は真ロールバック率と呼称される。真ロールバック率は応用背景下でデータ異常が出現する確率を反映する。

## 【0124】

4) 偽ロールバック率：P個のスケジュールにおいて、もしV個のスケジュールのうちの各スケジュールはいずれも偽ロールバックが存在すると仮定すれば、 $V/P$ は偽ロールバック率と呼称される。偽ロールバック率は並列トランザクションの並列実行能力を反映し、偽ロールバック率が低ければ低いほど並列性が高くなり、且つ計算リソースの利用率が高ければ高いほど、トランザクション並列効率が高くなる。

10

20

30

40

50

## 【0125】

以上、本願の実施例に関する基礎的概念を紹介し、以下、本願のトランザクション処理方法を説明する。本願の実施例においては、上記基礎的概念を理論基礎として、トランザクションとトランザクションとの間で構成された競合閉路グラフを利用してデータ異常識別を行い、トランザクションの実行過程において出現したデータ異常を効果的に識別することを可能にし、それによりトランザクション処理の効率を向上させる。

## 【0126】

表1に参照されるように、表1は本願の実施例に関したいくつかの符号、及び対応する意味を記述する。

## 【0127】

## 【表1】

表1

| 符号                                                             | 意味                                                                                                 |
|----------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| CCG                                                            | 競合に基づく有向閉路グラフであり、競合閉路グラフとも呼称される                                                                    |
| $s$                                                            | 1つのスケジュール                                                                                          |
| $s \doteq s'$                                                  | 2つのスケジュール $s$ と $s'$ の競合等価                                                                         |
| $TS(s)$                                                        | スケジュール $s$ におけるトランザクションの集合                                                                         |
| $VS(s)$                                                        | スケジュール $s$ におけるあるバージョン番号識別子変数付きの集合                                                                 |
| $O(s)$                                                         | 1つのスケジュールの操作集合                                                                                     |
| $O_i[x_m] O_j[x_n]$                                            | トランザクション $T_i$ 及び $T_j$ が同一変数 $x$ 上に作用する2つの操作であり、1つの競合と呼称され、ここで、1つの特例は $R_i[x] W_j[x]$ で表現することができる |
| $O_i O_j - O_j O_i$                                            | 2つの競合の間が組み合わせ、1つの閉路を構成し、データ異常を構成する必要条件であり、同時にデータ異常の形式化表現式でもある                                      |
| $x_m$                                                          | 変数 $x$ の $m$ 番目のバージョン値                                                                             |
| $R_i[x_m]$                                                     | トランザクション $T_i$ の読み取り操作 $R$ は変数 $x$ の $m$ 番目のバージョン値を読み取る                                            |
| $W_i[x_m]$                                                     | トランザクション $T_i$ の書き込み操作 $W$ は変数 $x$ の $m$ 番目のバージョン値を書込む                                             |
| $RW/WR/WW$                                                     | 3種類の競合の略記形式                                                                                        |
| $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_i[y_n] \cdots R_i[y_n]$ | (読み取り半順序) データ異常の形式化表現式                                                                             |

## 【0128】

上記記述された基礎的概念、及び表1に示された符号、及び対応する意味に基づいて、以下、図5を参照して、本願の実施例により提供されたトランザクション処理方法を説明する。図5は本願の実施例により提供されたトランザクション処理方法のフローチャートであり、図5に示すように、本願の実施例により提供されたトランザクション処理方法はデータベースシステムにおけるいずれか1つのノード機器に应用される。データベースシステムがスタンドアロンデータベースシステムであるときに、該ノード機器はスタンドアロン機器であってもよく、データベースシステムが分散型データベースシステムであるときに、該ノード機器は協調ノード機器、又はデータノード機器であってもよく、本願の実施例はステップ501、及びステップ502を含み、以下、各ステップをそれぞれ説明する。

## 【0129】

ステップ501：ノード機器はターゲットトランザクションの並列トランザクションを決定し、並列トランザクションと該ターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含み、ターゲットトランザクションはコミットすべきトランザクションである。

#### 【0130】

本願の実施例において、該ターゲットトランザクションはグローバルトランザクション、又は局所トランザクションである。ここで、該グローバルトランザクションとはノード間操作に関するトランザクションを指し、グローバルトランザクションは分散型トランザクションとも呼称され、局所トランザクションとは単一のノード操作にのみ関するトランザクションを指し、局所トランザクションはローカルトランザクションとも呼称される。本願の実施例はターゲットトランザクションのタイプについて限定しない。ここでは、読み取り書き込み操作とは、読み取り操作、及び書き込み操作のうちの少なくとも一種を指す。

#### 【0131】

ここで、ノード機器はターゲットトランザクションを実行するときに、トランザクション読み取りセット、及び書き込みセットにより、少なくとも1つの候補並列トランザクションをスクリーニングし、それによりスクリーニングされた少なくとも1つの候補並列トランザクションからターゲットトランザクションの並列トランザクションを決定する（並列トランザクションを決定するための具体的な実施形態は後続で紹介される）。ここでは、ターゲットトランザクションは端末により開始することができ、このときに、端末とノード機器はターゲットトランザクションを処理することに用いられるセッションを確立し、端末はノード機器にターゲットトランザクションの実行要求を送信し、ノード機器はターゲットトランザクションの実行要求に応答して、該ターゲットトランザクションを実行し始める。いくつかの実施例において、もし端末とノード機器は既に1つのセッションを確立したことがあれば、両者の間に新しいセッションを確立する必要がなく、既に確立されていた現在のセッションを多重化すればよい。

#### 【0132】

ステップ502：ノード機器はターゲットトランザクションの読み取りセットと並列トランザクションの書き込みセットとの間の第1共通集合を取得し、且つターゲットトランザクションの書き込みセットと並列トランザクションの読み取りセットとの間の第2共通集合を取得する。第1共通集合と第2共通集合のうちの少なくとも一項が空でないデータセットであり、且つターゲットトランザクションが並列トランザクションと競合するときに、ターゲットトランザクションと並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定し、ターゲット変数リストはターゲットトランザクションと並列トランザクションが作用した同一変数を含む。

#### 【0133】

本願の実施例において、ターゲットトランザクションが並列トランザクションと競合するとは、ターゲットトランザクションと並列トランザクションにおいて競合操作である少なくとも2つの操作が存在し、もしこのときに、ターゲットトランザクションをコミットすれば、データベースシステムにデータ異常が出現することを引き起こすようになる。ノード機器は上記第1共通集合と第2共通集合のうちの少なくとも一項が空でないデータセットであると決定し、且つターゲットトランザクションが並列トランザクションと競合するときに、該バージョン変更回数、及び該ターゲット変数リストに基づいて、データ異常タイプを決定し且つデータ異常タイプを報告する。ここで、空でないデータセットとは、空集合でないデータセットを指し、すなわち集合要素を含むデータセットである。

#### 【0134】

本願の実施例により提供されたトランザクション処理方法において、ノード機器はターゲットトランザクションの並列トランザクションを決定した後に、該ターゲットトランザクションの読み取りセットと該並列トランザクションの書き込みセットとの間の第1共通

10

20

30

40

50

集合、及び該ターゲットトランザクションの書き込みセットと該並列トランザクションの読み取りセットとの間の第2共通集合のうち空でない少なくとも一項が存在する場合に、ターゲットトランザクションが並列トランザクションと競合するか否かに対して判断を行う。もしターゲットトランザクションが並列トランザクションと競合すれば、この2つのトランザクションが作用した同じ変数、及び同じ変数のバージョン変更回数に従って、データ異常タイプを決定する。このような方式はターゲットトランザクションの実行過程において、ターゲットトランザクションが並列トランザクションと本格的に競合を構成するか否かを確認することを可能にし、且つ、2つのトランザクションが作用した同じ変数によりデータ異常タイプを決定することで、直列化可能理論に基づいて、データベースシステムにおける様々なデータ異常を識別することを可能にし、データ状態の整合性を保証し、データベースシステムのトランザクション処理効率も向上させる。

10

#### 【0135】

上記図5は本願により提供されたトランザクション処理方法に対する簡単な説明であり、以下、図6を組み合わせ、本願の実施例により提供されたトランザクション処理方法を詳細に紹介する。図6は本願の実施例により提供された他のトランザクション処理方法のフローチャートであり、図6に示すように、本願の実施例により提供されたトランザクション処理方法は、データベースシステムにおけるいずれか1つのノード機器に応用され、データベースシステムがスタンドアロンデータベースシステムであるときに、該ノード機器はスタンドアロン機器であってもよく、データベースシステムが分散型データベースシステムであるときに、該ノード機器は協調ノード機器、又はデータノード機器であってもよく、本願の実施例は以下ステップ601～ステップ609を含んでおり、以下、各ステップをそれぞれ説明する。

20

#### 【0136】

ステップ601：ノード機器はターゲットトランザクションの実行指令に応答して、ターゲットトランザクションの読み取りセット、及び書き込みセットを初期化する。

#### 【0137】

本願の実施例において、ノード機器はターゲットトランザクションの実行指令に応答して、該ターゲットトランザクションの読み取りセット、及び書き込みセットを空集合に初期化し、且つターゲットトランザクションにグローバルに一意的なトランザクションID、すなわち、トランザクション識別子を割り当てる。ここでは、異なるトランザクションのトランザクションIDはノード機器が相応な実行指令を受信する時間順序に応じて整数的に逓増し、たとえば、ノード機器はt1時点にターゲットトランザクションの実行指令を受信し、該ターゲットトランザクションのトランザクションIDを1000と命名し、もしノード機器はt2時点に他のトランザクションの実行指令を受信すれば、ノード機器は該他のトランザクションのトランザクションIDを1001と命名するが、本願の実施例はこれについて限定しない。

30

#### 【0138】

本願の実施例において、該初期化過程において、ノード機器はデータベースシステムが起動するときに、ノード機器のオペレーティングシステムは1つの内部メモリ空間を割り当てる。該内部メモリ空間は少なくとも1つのトランザクションの読み取りセット、及び書き込みセットを維持することに用いられ、ターゲットトランザクションを実行し始めるときに、ノード機器は該内部メモリ空間中から1つの内部メモリを申請する。該内部メモリは該ターゲットトランザクションの読み取りセット、及び書き込みセットを管理することに用いられ、それによりノード機器上にターゲットトランザクションの読み取りセット、及び書き込みセットの新規作成を完了し、且つ、新規作成した読み取りセット、及び書き込みセットをいずれも空集合に初期化する。

40

#### 【0139】

以下、ターゲットトランザクションの読み取りセット、及び書き込みセットの更新戦略を紹介する。

#### 【0140】

50

読み取りセット、及び書き込みセットの初期化を完了した後に、ノード機器はターゲットトランザクションの実行ステートメントに従って、該ターゲットトランザクションが変数に対する更新操作に関するか否かを決定する。変数に対する更新操作に関すると決定するときに、ノード機器は該ターゲットトランザクションに回答していずれか1つの変数を更新し、該変数を該ターゲットトランザクションの書き込みセット中に追加し、且つ該変数にバージョン番号を割り当てる。たとえば、あるトランザクションTは変数xを更新するときに、変数xをトランザクションTの書き込みセット中に追加し、且つ変数xにバージョン番号を割り当てる。異なるトランザクションの同一変数に対する書き込み操作のバージョン番号は整数的に逓増し、このときに、ノード機器は読み取り操作を実行すると、変数のバージョン番号を取得することができる。すなわち、データベースシステムは各変数に1つの逓増するバージョン番号を維持する。また、ノード機器はターゲットトランザクションの実行ステートメントに従って、いずれか1つの変数を読み取るときに、読み取られた変数は最新で且つリード・コミテッド規則を満たす変数である。

10

#### 【0141】

上記更新戦略はターゲットトランザクションの読み取りセット、及び書き込みセットがトランザクション操作の行いに伴ってリアルタイムに維持されると理解することもできる。もし読み取り操作が発生すれば、読み取られた変数が読み取りセットに入り、もし書き込み操作が発生すれば、読み取られた変数が書き込みセットに入る。同一トランザクションは同一変数を複数回書き込むと、複数の異なる新しいバージョンを有し、書き込みセットにおいて最新の変数バージョンで古い変数バージョンを代替する。

20

#### 【0142】

説明する必要がある点としては、本ステップ601は、すなわち、上記ステップ501において関するトランザクション読み取りセット、及び書き込みセット形成過程であり、読み取り書き込み段階におけるトランザクション読み取り書き込み集合の形成ステップである。

#### 【0143】

ステップ602：ノード機器はターゲットトランザクションにおける各操作の実行指令に回答して、ターゲットトランザクションの少なくとも1つの候補並列トランザクションを取得する。

#### 【0144】

本願の実施例において、ターゲットトランザクションの各々の操作（該操作は読み取り操作であってよく、さらに書き込み操作であってもよい）に対して、該操作の発生時に、準備段階に入り、ノード機器が臨界領域に入り、該ターゲットトランザクションの読み取りセット、及び書き込みセットをトラバーサルし、ターゲットトランザクションの少なくとも1つの候補並列トランザクション（たとえば、すべての並列トランザクション）のトランザクションIDを取得し、且つ該少なくとも1つの候補並列トランザクションのトランザクションIDを1つのタイプスクリプト（Type script、TS）中に記憶する。

30

#### 【0145】

本願の実施例において、ノード機器が少なくとも1つの候補並列トランザクションを取得する過程において、内部メモリにおける各変数上にはいずれも該変数を読み取り書き込みしたがコミットしていないトランザクションのトランザクションID（アクティブトランザクションIDと略称する）が記載されている。各変数上に記載されたアクティブトランザクションIDは1つであってもよく、複数であってもよく、さらに空であってもよい。ノード機器は読み取りセット、及び書き込みセットにおける各々の変数上に記載された各々のアクティブトランザクションIDを取得することにより、該少なくとも1つの候補並列トランザクションのトランザクションIDを得ることができる。

40

#### 【0146】

ステップ602は、ノード機器がターゲットトランザクションの各操作の発生時に、準備作業を始め、ターゲットトランザクションの準備段階に入ると理解することもできる。

50

ここで、準備段階はトランザクションTがコミットすることができるか否かのためにトランザクション並列アクセス制御異常認証を行う前の準備作業をきちんと行い、ノード機器は臨界領域において少なくとも1つの候補並列トランザクションのトランザクションIDを見つけた後に、臨界領域からログアウトする。

【0147】

説明する必要がある点としては、ステップ602は上記ステップ501における準備段階でもあり、少なくとも1つの候補並列トランザクションをスクリーニングすることに用いられる。

【0148】

ステップ603：ノード機器は少なくとも1つの候補並列トランザクションからターゲットトランザクションの並列トランザクションを決定し、並列トランザクションとターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含む。

【0149】

本願の実施例において、ノード機器はターゲットトランザクションの各操作の発生時に、該ターゲットトランザクションの少なくとも1つの候補並列トランザクションを取得し、少なくとも1つの候補並列トランザクションのうちのいずれか1つの候補並列トランザクションに対して、ノード機器は該候補並列トランザクション、及びターゲットトランザクションの読み取りセット、及び書き込みセットに従って、該候補並列トランザクションがターゲットトランザクションの並列トランザクションであるか否かを判断する。もし該候補並列トランザクションとターゲットトランザクションとの間に同一変数に作用する読み取り書き込み操作が含まれれば、該候補並列トランザクションを該ターゲットトランザクションの並列トランザクションとして決定し、ステップ604～ステップ607を実行し、そうでなければ、該候補並列トランザクションをタイプスクリプトTSのテールに改めて加える。

【0150】

ここで、ノード機器は直列化可能理論に基づいて、該少なくとも1つの候補並列トランザクションとターゲットトランザクションとの間に競合閉路グラフを構成するか否かで、整合性検査を行う。以下、タイプスクリプトTSにおけるいずれか1つの候補並列トランザクションを例として、本ステップ603の実施過程を詳細に説明する。ここでは、ステップ603の実施過程はステップ6031～ステップ6035を含み、以下、各ステップをそれぞれ説明する。

【0151】

ステップ6031：ノード機器はターゲットトランザクションの読み取りセットと候補並列トランザクションの書き込みセットとの間の第1候補共通集合、ターゲットトランザクションの書き込みセットと候補並列トランザクションの読み取りセットとの間の第2候補共通集合、及びターゲットトランザクションの書き込みセットと候補並列トランザクションの書き込みセットとの間の第3候補共通集合を取得する。

【0152】

ここで、ターゲットトランザクションがTであり、候補並列トランザクションがTLであるときに、ノード機器はターゲットトランザクションTと該候補並列トランザクションTLとの間の有向辺の数量を0に設定し、ターゲットトランザクション、及び候補並列トランザクションが作用した同じ変数のバージョン変更回数を0に設定し、ターゲット変数リストを空集合に設定する。ここで、該有向辺の数量はターゲットトランザクションと該候補並列トランザクションとの間が競合するか否かを表すことに用いられ、該バージョン変更回数はターゲットトランザクション、及び該候補並列トランザクションが作用した同じ変数のデータ状態を表すことに用いられる。該ターゲット変数リストはターゲットトランザクション、及び該候補並列トランザクションが作用した同じ変数を記憶することに用いられる。

【0153】

本願の実施例において、ノード機器は動的辺交差値 (Dynamic Line Int

10

20

30

40

50

ersection、DLI)で有向辺の数量を表し、変数状態値(Variable State、VS)でバージョン変更回数を表し、変数リスト(Variable List、VL)でターゲット変数リストを表し、すなわち、 $DLI=0$ 、 $VS=0$ 、 $VL=$ 空集合とする。

【0154】

本願の実施例において、第1候補共通集合は「 $S1=DSR(T) \cap DSW(TL)$ 」として表され、第2候補共通集合は「 $S2=DSR(TL) \cap DSW(T)$ 」として表され、第3候補共通集合は「 $S3=DSW(T) \cap DSW(TL)$ 」として表される。

【0155】

本願の実施例において、ノード機器は第1候補共通集合、第2候補共通集合、及び第3候補共通集合を取得するときに、ハッシュテーブルを使用してターゲットトランザクション、及び候補並列トランザクションのそれぞれの読み取り書き込み集合を記憶することができ、それにより線形時間複雑度内で対応する共通集合を得ることを可能にする。

【0156】

ステップ6032：もし第1候補共通集合、第2候補共通集合、及び第3候補共通集合の間の和集合が空集合であれば、ノード機器は候補並列トランザクションをタイプスクリプトTSのテールに改めて加え、もし第1候補共通集合、第2候補共通集合、及び第3候補共通集合のうち空集合でない少なくとも一項が存在すれば、ノード機器は以下のステップ6033～ステップ6035を実行する。

【0157】

ここで、もし第1候補共通集合、第2候補共通集合、及び第3候補共通集合の間の和集合が空集合であれば(すなわち、第1共通集合、第2共通集合、及び第3共通集合はいずれも空集合である)、該候補並列トランザクションはターゲットトランザクションの並列トランザクションではないということを説明している。もし第1候補共通集合、第2候補共通集合、及び第3候補共通集合のうち空集合でない少なくとも一項が存在すれば、該候補並列トランザクションはターゲットトランザクションの並列トランザクションであるということを説明している。ノード機器は、該候補並列トランザクションをターゲットトランザクションの並列トランザクション、すなわち、本ステップ603に関する並列トランザクションとして決定し、以下のステップ6033～ステップ6035を実行する。また、ノード機器は候補並列トランザクションがターゲットトランザクションの並列トランザクションであると決定するときに、第1候補共通集合はすなわち第1共通集合であり、第2候補共通集合はすなわち第2共通集合であり、第3候補共通集合はすなわち第3共通集合である。

【0158】

以下、該候補並列トランザクションは決定された並列トランザクションであることを例として、ステップ6033～ステップ6035を説明する。

【0159】

ステップ6033：もし該第1共通集合が空集合でなければ、ノード機器は有向辺の数量に対して増分処理を行い、且つ第1共通集合における変数をターゲット変数リスト中に追加し、もし第1共通集合中にバージョンが異なる変数が存在すれば、ノード機器はバージョン変更回数に対して増分処理を行う。

【0160】

ここで、増分処理とは、対応する値を1増やすことを指す。例示的には、有向辺の数量がDLIであり、バージョン変更回数がVSであることを例として、上記2種類の増分処理はそれぞれ $DLI=DLI++$ 及び $VS=VS++$ として表される。また、ノード機器が該第1共通集合における変数を該ターゲット変数リスト中に追加することとは、予め設定した順序に応じて、変数を該ターゲット変数リスト中に秩序よく追加することを指す。

【0161】

ステップ6034：もし第2共通集合が空でなければ、ノード機器は有向辺の数量に対して増分処理を行い、第2共通集合における変数をターゲット変数リスト中に追加し、も

10

20

30

40

50

し第2共通集合中にバージョンが異なる変数が存在すれば、ノード機器はバージョン変更回数に対して増分処理を行う。

【0162】

ここで、ステップ6034と上記ステップ6033は実現過程上において類似しており、従って、ここで再度繰り返し記述しない。

【0163】

ステップ6035：もし第3共通集合が空集合でなければ、ノード機器は並列トランザクションのコミット状態に基づいて、データ異常タイプを決定し、もし第3共通集合が空集合であれば、ノード機器は後続のステップ604を実行する。

【0164】

ここで、もし第3共通集合が空集合でなければ、ターゲットトランザクションと並列トランザクションとの間に書き込み書き込み競合が出現したことを説明している。書き込み書き込み競合はデータ整合性状態モデル下でデータエラーを引き起こす原因の1つであり、すなわちダーティライトデータ異常を禁止する。従って、このような場合に、ノード機器は該並列トランザクションのコミット状態に基づいて、データ異常タイプを決定し、且つ書き込み書き込み異常の発生をレポートし、現在のプロセスを終了し、ターゲットトランザクションに対してロールバックを行う。第3共通集合が空集合であるときに、書き込み書き込み競合が発生していないことが表明され、それによりノード機器は後続のステップ604を実行する。

【0165】

いくつかの実施例において、もし該並列トランザクションがコミットされておらず且つ該ターゲットトランザクションのターゲットパラメータ $T.no\_committed$ が1であれば、ノード機器はデータ異常タイプがダーティライト異常であると決定する。ここで、該ターゲットパラメータ $T.no\_committed$ は該ターゲットトランザクションの読み取り書き込み集合と対応するコミット済みトランザクションの数量を表すことに用いられる。サイクル過程において、もし前の並列トランザクションとターゲットトランザクションとの間にデータ異常が存在しなければ、今回のサイクルを実行する前に、前の並列トランザクションをターゲットトランザクションと合併する必要がある、且つ合併した後に得られた新しいトランザクションを今回サイクルのターゲットトランザクションとして決定して、合併は2つのトランザクションの読み取り書き込み集合を合併することにより実現することができる。従って、いずれか1回のサイクルにおいて、ターゲットトランザクションの読み取り書き込み集合は合併された後の読み取り書き込み集合である可能性があり、ターゲットトランザクションの読み取り書き込み集合中に既にコミットされたトランザクションの読み取り書き込み集合の成分を含む場合に、ターゲットパラメータ $T.no\_committed$ は読み取り書き込み集合における対応するコミット済みトランザクションの数量を記述することに用いられる。

【0166】

いくつかの実施例において、もし該並列トランザクションがコミットされ且つ該第1共通集合と該第3共通集合との間の共通集合が空集合でなければ、データ異常タイプが紛失更新異常であると決定する。

【0167】

例示的には、第1共通集合、第2共通集合、及び第3共通集合がそれぞれ $S1$ 、 $S2$ 、及び $S3$ として表されることを例として、もし $S3 \neq [\text{数}2]$ であれば、以下のいずれか1つの条件を満たすときに、ノード機器はデータ異常タイプを決定し、書き込み書き込み異常の発生をレポートし、サイクルを終止させる。ここで、満たすべき条件は、1) 該並列トランザクションのコミットが完了しておらず且つ該ターゲットトランザクションのターゲットパラメータ $T.no\_committed = 1$ であり、このときに、書き込み書き込み競合を構成し、データ異常タイプがダーティライト異常であることと、2) 該並列トランザクションが既にコミットされ、且つ $S1 \cap S3$ が空集合でなく、このときに、データ異常タイプが紛失更新異常であることと、を含む。

10

20

30

40

50

【0168】

【数2】

 $\emptyset$ 

【0169】

説明する必要がある点としては、ノード機器は上記ステップ6031～ステップ6035を同期に実行してもよく、さらに任意の順序に応じて上記ステップ6031～ステップ6035を実行してもよく、本願の実施例はこれについて限定しない。

10

【0170】

上記ステップ603により、ノード機器は少なくとも1つの候補並列トランザクションからターゲットトランザクションの並列トランザクションを決定し、もしターゲットトランザクションと並列トランザクションとの間の第3共通集合が空集合であれば、ノード機器は以下のステップ604を実行する。

【0171】

ステップ604：もし第1共通集合と第2共通集合のうち空集合でない少なくとも一項が存在し、且つターゲットトランザクションが並列トランザクションと競合すれば、ノード機器はターゲットトランザクションと並列トランザクションが作用した同じ変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定し、ターゲット変数リストはターゲットトランザクションと並列トランザクションが作用した同じ変数を含む。

20

【0172】

ここで、該第1共通集合と該第2共通集合のうち、空集合でない少なくとも一項が存在することは以下の3種類の場合を含む。1) 第1共通集合が空集合でないが、第2共通集合が空集合である。2) 第1共通集合が空集合であるが、第2共通集合が空集合でない。3) 第1共通集合が空集合でなく、第2共通集合も空集合でない。もし第1共通集合と第2共通集合が上記3種類の場合のうちのいずれかの一種を満たし、且つ該ターゲットトランザクションが該並列トランザクションと競合すれば、ノード機器はバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定する。ここで、ノード機器はターゲットトランザクションと並列トランザクションとの間の有向辺の数量に基づいて、ターゲットトランザクションが並列トランザクションと競合するか否か決定し、すなわち、もしターゲットトランザクションと該並列トランザクションとの間の有向辺の数量が2以上であれば、ターゲットトランザクションが並列トランザクションと競合すると決定する。

30

【0173】

以下、ステップ604におけるデータ異常タイプの決定過程を説明する。まず、以下の4つの点により、本願の実施例に関するデータ異常タイプの所属カテゴリ、サブカテゴリ、及び隔離レベルをそれぞれ紹介する。

【0174】

第1点としては、データ異常タイプの所属カテゴリである。

40

【0175】

本願の実施例において、データ異常タイプの所属カテゴリは書き込み異常カテゴリ (Write Anomaly Type、WAT)、読み取り異常カテゴリ (Read Anomaly Type、RAT)、及び交差異常カテゴリ (Intersect Anomaly Type、IAT) を含む。

【0176】

ここで、もし1つのスケジュール中に異なるバージョンの同一変数に作用する少なくとも2つ (第1指定数量と呼称される) の書き込み操作が存在すれば (書き込み異常条件を満たすと呼称される)、該データ異常タイプの所属カテゴリは書き込み異常カテゴリであ

50

り、もし1つのスケジュール中に異なるバージョンの同一変数に作用する少なくとも2つの書き込み操作が存在せず（書き込み異常条件を満たさないと呼称される）、且つ該スケジュール中に同じバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在すれば（読み取り異常条件を満たすと呼称される）、該データ異常タイプの所属カテゴリは読み取り異常カテゴリである。もし1つのスケジュール中に異なるバージョンの同一変数に作用する少なくとも2つの書き込み操作が存在せず（書き込み異常条件を満たさないと呼称される）、且つ該スケジュール中に同じバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在せず（読み取り異常条件を満たさないと呼称される）、且つ該スケジュール中に異なるバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在すれば（交差異常条件を満たすと呼称される）、該データ異常タイプの所属カテゴリは交差異常カテゴリである。

10

【0177】

以下、上記表1に示された符号、及び対応する意味を組み合わせ、1つのスケジュールsに対応するCCGグラフにおいて $\exists T_i, T_j \in TS(s)$ 、且つ $i \neq j$ であれば、データ異常タイプの所属カテゴリは以下の3種類の形式として表すことができ、以下、それぞれ説明する。

【0178】

1、書き込み異常カテゴリ。

【0179】

もし $\exists \langle T_i, T_j \rangle \in E = W_i[x_m] W_j[x_{m+1}]$ であれば、該スケジュールsにデータ異常が存在し、且つ該データ異常タイプの所属カテゴリは書き込み異常カテゴリである。

20

【0180】

2、読み取り異常カテゴリ。

【0181】

もし $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m] W_j[x_{m+1}]$ であり、且つ $\exists \langle T_i, T_j \rangle \in E = W_i[x_m] R_j[x_m]$ であれば、該スケジュールsにデータ異常が存在し、且つ該データ異常タイプの所属カテゴリは読み取り異常カテゴリである。

【0182】

3、交差異常カテゴリ。

30

【0183】

もし $\exists \langle T_i, T_j \rangle \in E \neq W_i[x_m] W_j[x_{m+1}]$ であり、且つ $\forall \langle T_i, T_j \rangle \in E \neq W_i[x_m] R_j[x_m]$ であり、且つ $\exists \langle T_i, T_j \rangle \in E = R_i[x_m] W_j[x_{m+1}]$ であれば、該スケジュールsにデータ異常が存在し、且つ該データ異常タイプの所属カテゴリは交差異常カテゴリである。

【0184】

第2点としては、データ異常タイプのサブカテゴリである。

【0185】

本願の実施例において、第1点において競合に従う差異が記述されており、データ異常タイプを書き込み異常カテゴリ、読み取り異常カテゴリ、及び交差異常カテゴリの3種類に分ける。ここでは、データ異常識別の精度を向上させるために、データ異常のタイプを継続的に区分して、サブカテゴリを得る。

40

【0186】

ここで、もし1つのデータ異常が同一変数上に発生すれば、データ異常タイプの所属カテゴリにおけるサブカテゴリはシングルメタデータ異常（Single-Meta Data Anomalies、SDA）に属し、もし1つのデータ異常が2つ（第3指定数量と呼称される）の変数上に発生すれば、データ異常タイプの所属カテゴリにおけるサブカテゴリはダブルメタデータ異常（Double-Meta Data Anomalies、DDA）に属し、もし1つのデータ異常が上記2種類のサブカテゴリに属さなければ、データ異常タイプの所属カテゴリにおけるサブカテゴリはマルチメタデータ異常（Mu

50

lti-Meta Data Anomalies、MDA) に属する。

【0187】

例示的には、1つのデータ異常(Data Anomalies、DA、daと略記する)における変数の個数はNVとして表され、トランザクションの個数はNTとして表されると仮定し、NVとNTとの組み合わせをデータ異常の特徴と呼称し、Cda=(NV, NT)と記せば、上記データ異常タイプのサブカテゴリは以下の3種類の場合として表され、以下、それぞれ説明する。

【0188】

1、シングルメタデータ異常。

【0189】

もしCda=(1, 2)であれば、データ異常が2つのトランザクションの1つの変数上に発生し、この場合、データ異常タイプの所属カテゴリにおけるサブカテゴリはシングルメタデータ異常に属する。

【0190】

2、ダブルメタデータ異常。

【0191】

もしCda=(2, 2)であれば、データ異常が2つのトランザクションの2つの変数上に発生し、この場合、データ異常タイプの所属カテゴリにおけるサブカテゴリはダブルメタデータ異常に属する。

【0192】

3、マルチメタデータ異常。

【0193】

もしCda≠(1, 2)、且つCda≠(2, 2)であれば、データ異常タイプの所属カテゴリにおけるサブカテゴリはマルチメタデータ異常に属する。

【0194】

第3点としては、データ異常タイプの細分類である。

【0195】

本願の実施例において、上記第1点(データ異常タイプの所属カテゴリ)、及び第2点(データ異常タイプのサブカテゴリ)に示された分類に応じて、細分類された各種のデータ異常の所属カテゴリ、名称、対応する形式化表現、対応するCCGグラフ、及び対応するサブカテゴリが与えられる。具体的には、表2、及び図7～図9に示すとおりである。ここで、図7は本願の実施例により提供されたシングルメタデータ異常CCGグラフの模式図である。図8は本願の実施例により提供されたダブルメタデータ異常CCGグラフの模式図である。図9は本願の実施例により提供されたマルチメタデータ異常CCGグラフの模式図である。

【0196】

10

20

30

40

50

【表 2】

表 2

| 所属カ<br>テゴリ | データ異常<br>の名称     | 形式化表現                                                                                                                                                              | C C G        | サブカ<br>テゴリ |
|------------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|------------|
| W A T      | 紛失更新異常           | $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_i[x_{m+2}]$                                                                                                                 | 図 7 における (b) | S D A      |
|            | 紛失自己更新異常         | $W_i[x_m] \cdots W_j[x_{m+1}] \cdots R_i[x_{m+1}]$                                                                                                                 | 図 7 における (c) | S D A      |
|            | 全書き込み異常          | $W_i[x_m] \cdots W_j[x_{m+1}] \cdots W_i[x_{m+2}]$                                                                                                                 | 図 7 における (d) | S D A      |
|            | 読み取り書き込み半順序異常 1  | $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                 | 図 8 における (f) | D D A      |
|            | 読み取り書き込み半順序異常 2  | $W_i[x_m] \cdots W_j[x_{m+1}] \cdots R_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                 | 図 8 における (g) | D D A      |
|            | リピータブルライト半順序異常 1 | $W_i[x_m] \cdots W_j[x_{m+1}] \cdots W_j[y_n] \cdots R_i[y_n]$                                                                                                     | 図 8 における (h) | D D A      |
|            | リピータブルライト半順序異常 2 | $W_i[x_m] \cdots R_j[x_m] \cdots W_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                     | 図 8 における (i) | D D A      |
|            | 全書き込み半順序異常       | $W_i[x_m] \cdots W_j[x_{m+1}] \cdots W_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                 | 図 8 における (c) | D D A      |
|            | 階段式書き込み異常        | $\cdots W_i[x_m] \cdots W_j[x_{m+1}] \cdots$                                                                                                                       | 図 9 における (a) | M D A      |
| R A T      | ノンリピータブルリード異常    | $R_i[x_m] \cdots W_j[x_{m+1}] \cdots R_i[x_{m+1}]$                                                                                                                 | 図 7 における (a) | S D A      |
|            | 中間読み取り異常         | $W_i[x_m] \cdots R_j[x_m] \cdots W_i[x_{m+1}]$                                                                                                                     | 図 7 における (e) | S D A      |
|            | 読み取り半順序異常 1      | $R_i[x_m] \cdots W_j[x_{m+1}] \cdots W_j[y_n] \cdots R_i[y_n]$                                                                                                     | 図 8 における (d) | D D A      |
|            | 読み取り半順序異常 2      | $W_i[x_m] \cdots R_j[x_m] \cdots R_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                     | 図 8 における (e) | D D A      |
|            | 書き込み読み取り半順序異常    | $W_i[x_m] \cdots R_j[x_m] \cdots W_j[y_n] \cdots W_i[y_n]$                                                                                                         | 図 8 における (b) | D D A      |
|            | 階段式読み取り異常        | $\cdots W_i[x_m] \cdots R_j[x_m] \cdots$ 且つ $\cdots W_{i i}[x_m] \cdots W_{j j}[x_{m+1}] \cdots$ を含まない                                                             | 図 9 における (b) | M D A      |
| I A T      | 書き込み半順序異常        | $R_i[x_m] \cdots W_j[x_{m+1}] \cdots R_j[y_n] \cdots W_i[y_{n+1}]$                                                                                                 | 図 8 における (a) | D D A      |
|            | 階段式交差異常          | $\cdots R_i[x_m] \cdots W_j[x_{m+1}] \cdots$ 且つ $\cdots W_{i i}[x_m] \cdots W_{j j}[x_m] \cdots$ 及び $\cdots W_{i i i}[x_m] \cdots W_{j j j}[x_{m+1}] \cdots$ を含まない | 図 9 における (c) | M D A      |

## 【0197】

説明する必要がある点としては、図 7～図 9 に示された C C G グラフの模式図の関連する記述については、図 2～図 4 に対応する記述段落を参照すればよく、本願の実施例はここで再度繰り返し記述しない。

## 【0198】

説明する必要がある点としては、シングルメタデータ異常、及びダブルメタデータ異常に比べて、マルチメタデータ異常の発生確率が比較的小さい。以下、図 9 に示されたマルチメタデータ異常 C C G グラフに対応するスケジュール表現式を例示的に説明する。

## 【0199】

図 9 における (a) グラフに示すように、競合閉路グラフ 9-1 においては階段式書き込み異常であり、階段式書き込み異常のスケジュール表現式は  $s = R_1 [x_0] W_2 [x_1] W_2 [y_1] W_3 [y_2] R_3 [z_0] W_1 [z_1]$  であり、該スケジュール中には WW 競合が含まれる。

【0200】

図 9 における (b) グラフに示すように、競合閉路グラフ 9-2 においては階段式読み取り異常であり、階段式読み取り異常のスケジュール表現式は  $s = R_1 [x_0] W_2 [x_1] W_2 [y_1] R_3 [y_1] R_3 [z_0] W_1 [z_1]$  であり、該スケジュール中には WR 競合が含まれるが、WW 競合が含まれていない。

【0201】

図 9 における (c) グラフに示すように、競合閉路グラフ 9-3 においては階段式交差異常であり、階段式交差異常のスケジュール表現式は  $s = R_1 [x_0] W_2 [x_1] R_2 [y_0] W_3 [y_1] R_3 [z_0] W_1 [z_1]$  であり、該スケジュール中には RW 競合が含まれるが、WW 競合、及び WR 競合が含まれていない。

【0202】

第 4 点としては、データ異常タイプの隔離レベルである。

【0203】

本願の実施例において、上記第 3 点に示されたデータ異常の細分類に応じて、2 種類の隔離レベルを提供しており、それぞれ読み取り書き込み異常なし (No Read-Write Anomalies、NRW)、及び任意の異常なし (No Anomalies、NA) である。ここで、隔離レベルの等級付けの根拠は、データ異常の出現の許可程度であり、隔離レベルにより、並列アクセス制御処理の複雑度を低減させることを可能にする。ここで、NRWとは、WAT及びRAT類のデータ異常の出現を許可しないが、IAT類データ異常を除去できないことを指し、NAとは、任意のデータ異常の発生を許可しないことを指し、ANSI-SQL標準により定義された直列化可能隔離レベルと等価である。従って、NRWレベルはNAレベルよりも低い。

【0204】

以下、表 3 に参照されるように、本願の実施例により提供された隔離レベルを記述する。

【0205】

10

20

30

40

50

【表 3】

表 3

| カテゴリ  | データ異常の名称         | N R W | N A   |
|-------|------------------|-------|-------|
| W A T | 紛失更新異常           | 許可しない | 許可しない |
|       | 紛失自己更新異常         | 許可しない | 許可しない |
|       | 全書き込み異常          | 許可しない | 許可しない |
|       | 読み取り書き込み半順序異常 1  | 許可しない | 許可しない |
|       | 読み取り書き込み半順序異常 2  | 許可しない | 許可しない |
|       | リピータブルライト半順序異常 1 | 許可しない | 許可しない |
|       | リピータブルライト半順序異常 2 | 許可しない | 許可しない |
|       | 全書き込み半順序異常       | 許可しない | 許可しない |
|       | 階段式書き込み異常        | 許可しない | 許可しない |
|       |                  |       |       |
| R A T | ノンリピータブルリード異常    | 許可しない | 許可しない |
|       | 中間読み取り異常         | 許可しない | 許可しない |
|       | 読み取り半順序異常 1      | 許可しない | 許可しない |
|       | 読み取り半順序異常 2      | 許可しない | 許可しない |
|       | 書き込み読み取り半順序異常    | 許可しない | 許可しない |
|       | 階段式読み取り異常        | 許可しない | 許可しない |
| I A T | 書き込み半順序異常        | 許可    | 許可しない |
|       | 階段式交差異常          | 許可    | 許可しない |

## 【0206】

説明する必要がある点としては、ノード機器は表 3 において記述された隔離レベルの関連情報に基づいて、並列アクセス制御処理を決定する。

## 【0207】

本願の実施例において、表 3 において記述された隔離レベルに応じて、ノード機器は規則を設定することによりデータ異常の出現を回避して、N R Wレベルに達することを可能にする。たとえば、WW辺の出現を禁止する規則を採用してW A T異常の出現を回避し、また、たとえば、リード・コミテッド、スナップショット読み取り等方式を採用してR A T異常の出現を回避する。このようにすると、隔離レベルの階層個数を減少させ、並列アクセス制御処理の効率を向上させることを可能にする。たとえば、直列化可能スナップショット隔離（S e r i a l i z a b l e S n a p s h o t I s o l a t i o n、S S I）技術を採用して、連続的な 2 つの R W 辺に異常が出現するか否かを識別することにより、N A レベルに達することができ、従って、並列アクセス制御処理の効率を向上させることを可能にする。

## 【0208】

上記 4 つの点により、本願の実施例に関するデータ異常タイプの所属カテゴリ、サブカテゴリ、及び隔離レベルを記述し、以下、上記 4 つの点で記述された情報を基礎として、本ステップ 604 の実現過程を詳細に説明する。ここで、ステップ 604 はステップ 6041～ステップ 6043 を含み、以下、各ステップをそれぞれ説明する。

## 【0209】

ステップ 6041：ノード機器はターゲットトランザクションと並列トランザクションにおいて同一変数に作用する読み取り書き込み操作に基づいて、所属カテゴリを決定する。

## 【0210】

ここで、所属カテゴリとは、データ異常タイプが所属するカテゴリを指し、ステップ6041は以下の3種類の場合を含み、以下、該3種類の場合をそれぞれ説明する。

## 【0211】

場合1：もしターゲットトランザクションと該並列トランザクションにおいて異なるバージョンの同一変数に作用する第1指定数量（たとえば、2つである）の書き込み操作が存在すれば、ノード機器は所属カテゴリが書き込み異常カテゴリであると決定する。

## 【0212】

例示的には、該ターゲットトランザクションが $T_i$ であり、該並列トランザクションは $T_j$ であり、もし $\exists < T_i, T_j > \in E = W_i[x_m] W_j[x_{m+1}]$ であれば、ノード機器は所属カテゴリが書き込み異常カテゴリであると決定する。

10

## 【0213】

場合2：もし該ターゲットトランザクションと該並列トランザクションにおいて異なるバージョンの同一変数に作用する2つの書き込み操作が存在せず、且つ該ターゲットトランザクションと該並列トランザクションにおいて同じバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在すれば、ノード機器は所属カテゴリが読み取り異常カテゴリであると決定する。

## 【0214】

例示的には、該ターゲットトランザクションが $T_i$ であり、該並列トランザクションは $T_j$ であると仮定し、もし $\forall < T_i, T_j > \in E \neq W_i[x_m] W_j[x_{m+1}]$ であり、且つ $\exists < T_i, T_j > \in E = W_i[x_m] R_j[x_m]$ であれば、該ノード機器は所属カテゴリが読み取り異常カテゴリであると決定する。

20

## 【0215】

場合3：もし該ターゲットトランザクションと該並列トランザクションにおいて異なるバージョンの同一変数に作用する2つの書き込み操作が存在せず、且つ該ターゲットトランザクションと該並列トランザクションにおいて同じバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在せず、且つ該ターゲットトランザクションと該並列トランザクションにおいて異なるバージョンの同一変数に作用する読み取り操作、及び書き込み操作が存在すれば、ノード機器は所属カテゴリが交差異常カテゴリであると決定する。

30

## 【0216】

例示的には、該ターゲットトランザクションが $T_i$ であり、該並列トランザクションが $T_j$ であると仮定し、もし $\forall < T_i, T_j > \in E \neq W_i[x_m] W_j[x_{m+1}]$ であり、且つ $\forall < T_i, T_j > \in E \neq W_i[x_m] R_j[x_m]$ であり、且つ $\exists < T_i, T_j > \in E = R_i[x_m] W_j[x_{m+1}]$ であれば、ノード機器は所属カテゴリが交差異常カテゴリであると決定する。

## 【0217】

ステップ6042：ノード機器はデータ異常タイプの所属カテゴリ、及びターゲット変数リストに基づいて、所属カテゴリにおけるサブカテゴリを決定する。

## 【0218】

ここで、上記ステップ603から明らかなように、ターゲット変数リスト中には該ターゲットトランザクション、及び該並列トランザクションが作用した同じ変数が記憶されている。すなわち、ノード機器はターゲット変数リストに基づいて、該ターゲットトランザクション、及び該並列トランザクションが作用した同じ変数の数量を取得し、それにより上記3種類のサブカテゴリの記述を組み合わせ、所属カテゴリにおけるサブカテゴリを決定する。

40

## 【0219】

例示的には、本ステップ6042は以下の3種類の場合を含み、以下、該3種類の場合をそれぞれ説明する。

## 【0220】

50

場合 1：もし該ターゲット変数リストにおける変数の数量が第 2 指定数量（たとえば、1 つ）であれば、ノード機器は該所属カテゴリにおけるサブカテゴリがシングルメタデータ異常に属すると決定する。

【0221】

場合 2：もし該ターゲット変数リストにおける変数の数量が第 3 指定数量（たとえば、2 つ）であれば、ノード機器は該所属カテゴリにおけるサブカテゴリがダブルメタデータ異常に属すると決定する。

【0222】

場合 3：もし該ターゲット変数リストにおける変数の数量が第 4 指定数量であり、ここで、第 4 指定数量は、第 2 指定数量、及び第 3 指定数量の両方とは異なる、すなわち、該所属カテゴリにおけるサブカテゴリは該シングルメタデータ異常に属さないだけでなく該ダブルメタデータ異常にも属さなければ、ノード機器は該所属カテゴリにおけるサブカテゴリがマルチメタデータ異常に属すると決定する。

【0223】

ステップ 6043：ノード機器は所属カテゴリにおけるサブカテゴリ、バージョン変更回数、及び並列トランザクションとターゲットトランザクションにおいて同一変数に作用する読み取り書き込み操作に基づいて、データ異常タイプを決定する。

【0224】

ここで、上記ステップ 603 から明らかなように、バージョン変更回数は該ターゲットトランザクション、及び該並列トランザクションが作用した同じ変数のデータ状態を表すことに用いられる。また、上記 [表 2] から明らかなように、異なるデータ異常に対応するバージョン変更回数、及び 2 つのトランザクションが同一変数に作用する読み取り書き込み操作は異なり、従って、本ステップ 6043 において、ノード機器はバージョン変更回数、及び該並列トランザクションと該ターゲットトランザクションにおいて同一変数に作用する読み取り書き込み操作に基づいて、該データ異常タイプを決定する。

【0225】

例示的には、以下、8 種のシーンを例として、本ステップ 6043 の実施過程をそれぞれ説明する。

【0226】

シーン 1：もし所属カテゴリが書き込み異常カテゴリであり、該サブカテゴリがシングルメタデータ異常であれば、本ステップ 6043 は以下の 3 種類の場合を含み、以下、該 3 種類の場合をそれぞれ説明する。

【0227】

1) もし該バージョン変更回数が 1 であれば、ノード機器は該データ異常タイプが紛失自己更新異常であると決定する。

【0228】

2) もし該バージョン変更回数が 2 であり、該ターゲットトランザクションと該並列トランザクションにおいて同一変数に作用する第 5 指定数量（たとえば、3 つ）の書き込み操作が存在すれば、ノード機器は該データ異常タイプが全書き込み異常であると決定する。

【0229】

3) もし該バージョン変更回数が 2 であり、該ターゲットトランザクションと該並列トランザクションにおいて同一変数に作用する 2 つの書き込み操作、及び第 6 指定数量（たとえば、1 つ）の読み取り操作が存在すれば、ノード機器は該データ異常タイプが紛失更新異常であると決定する。

【0230】

シーン 2：もし該データ異常タイプの所属カテゴリが書き込み異常カテゴリであり、該サブカテゴリがダブルメタデータ異常であれば、本ステップ 6043 は以下の 3 種類の場合を含み、以下、該 3 種類の場合をそれぞれ説明する。

【0231】

1) もし該バージョン変更回数が 1 であり、該ターゲットトランザクションと該並列ト

10

20

30

40

50

ランザクションにおいて異なるバージョンの第1変数に作用する書き込み操作、及び読み取り操作が存在し（第1条件を満たすと呼称される）、及び異なるバージョンの第2変数に作用する2つの書き込み操作が存在すれば（第2条件を満たすと呼称される）、ノード機器は該データ異常タイプが読み取り書き込み半順序異常であると決定する。ここで、第1条件を満たすとは、ターゲット読み取り書き込み操作が異なるバージョンの第1変数に作用する書き込み操作、及び読み取り操作を含むことを指し、第2条件を満たすとは、ターゲット読み取り書き込み操作が異なるバージョンの第2変数に作用する第1指定数量の書き込み操作を含むことを指す。

【0232】

2) もし該バージョン変更回数が1であり、該ターゲットランザクションと該並列ランザクションにおいて同じバージョンの第1変数に作用する書き込み操作、及び読み取り操作が存在し（第3条件を満たすと呼称される）、及び異なるバージョンの第2変数に作用する2つの書き込み操作が存在すれば（第2条件を満たすと呼称される）、ノード機器は該データ異常タイプがリピータブルライト半順序異常であると決定する。ここで、第3条件を満たすとは、ターゲット読み取り書き込み操作が同一バージョンの第1変数に作用する書き込み操作、及び読み取り操作を含むことを指す。

10

【0233】

3) もし該バージョン変更回数が1であり、該ターゲットランザクションと該並列ランザクションにおいて異なるバージョンの第1変数に作用する2つの書き込み操作が存在し（第4条件を満たすと呼称される）、且つ異なるバージョンの第2変数に作用する2つの書き込み操作が存在すれば（第2条件を満たすと呼称される）、ノード機器は該データ異常タイプが全書き込み半順序異常であると決定する。ここで、第4条件を満たすとは、ターゲット読み取り書き込み操作が異なるバージョンの第1変数に作用する第1指定数量の書き込み操作を含むことを指す。

20

【0234】

シーン3：もし該データ異常タイプの所属カテゴリが読み取り異常カテゴリであり、該サブカテゴリがシングルメタデータ異常であれば、本ステップ6043は以下の2種類の場合を含み、以下、該2種類の場合をそれぞれ説明する。

【0235】

1) もし該バージョン変更回数が1であり、該ターゲットランザクションと該並列ランザクションにおいて同一変数に作用する2つの読み取り操作、及び1つの書き込み操作が存在すれば、ノード機器は該データ異常タイプがノンリピータブルリード異常であると決定する。

30

【0236】

2) もし該バージョン変更回数が1であり、該ターゲットランザクションと該並列ランザクションにおいて同一変数に作用する2つの書き込み操作、及び1つの読み取り操作が存在すれば、ノード機器は該データ異常タイプが中間読み取り異常であると決定する。

【0237】

シーン4：もし該データ異常タイプの所属カテゴリが読み取り異常カテゴリであり、該サブカテゴリがダブルメタデータ異常であれば、本ステップ6043は以下の2種類の場合を含み、以下、該2種類の場合をそれぞれ説明する。

40

【0238】

1) もし該バージョン変更回数が第7指定数量（たとえば、ゼロ）であれば、ノード機器は該データ異常タイプが書き込み読み取り半順序異常であると決定する。

【0239】

2) もし該バージョン変更回数が1であれば、ノード機器は該データ異常タイプが読み取り半順序異常であると決定する。

【0240】

シーン5：もし該データ異常タイプの所属カテゴリが交差異常カテゴリであり、該サブカテゴリがダブルメタデータ異常であれば、本ステップ6043は、もし該バージョン変

50

更回数が1であれば、ノード機器は該データ異常タイプが書き込み半順序異常であると決定することを含む。

【0241】

シーン6：もし該データ異常タイプの所属カテゴリが書き込み異常カテゴリであり、該サブカテゴリがマルチメタデータ異常であれば、本ステップ6043は、ノード機器は該データ異常タイプが階段式書き込み異常であると決定することを含む。

【0242】

シーン7：もし該データ異常タイプの所属カテゴリが読み取り異常カテゴリであり、該サブカテゴリがマルチメタデータ異常であれば、本ステップ6043は、ノード機器は該データ異常タイプが階段式読み取り異常であると決定することを含む。

【0243】

シーン8：もし該データ異常タイプの所属カテゴリが交差異常カテゴリであり、該サブカテゴリがマルチメタデータ異常であれば、本ステップ6043は、ノード機器は該データ異常タイプが階段式交差異常であると決定することを含む。

【0244】

説明する必要がある点としては、ノード機器はステップ6041～ステップ6043の順序に応じて実行し、すなわち、ノード機器はまず、データ異常タイプの所属カテゴリを決定し、次にサブカテゴリを決定し、最後に細分類されたデータ異常タイプを決定する。いくつかの実施例において、上記ステップ6041～ステップ6043は以下のステップ1～ステップ3に置換される。

【0245】

ステップ1：ノード機器は該ターゲット変数リストに基づいて、サブカテゴリを決定する。

【0246】

ステップ2：ノード機器は該ターゲットトランザクションと該並列トランザクションにおいて同一変数に作用する読み取り書き込み操作に基づいて、所属カテゴリを決定する。

【0247】

ステップ3：ノード機器は所属カテゴリにおけるサブカテゴリ、該バージョン変更回数、及び該並列トランザクションと該ターゲットトランザクションにおいて同一変数に作用する読み取り書き込み操作に基づいて、該データ異常タイプを決定する。

【0248】

すなわち、ノード機器はまず、データ異常タイプのサブカテゴリを決定し、次に所属カテゴリを決定し、最終的に細分類されたデータ異常タイプを決定する。

【0249】

いくつかの実施例において、ノード機器はさらにデータ異常タイプの所属カテゴリ、及びサブカテゴリを同期に決定することができ、本願の実施例はこれについて限定しない。

【0250】

上記ステップ604により、ノード機器はデータ異常タイプを決定した後に、以下のステップ605～ステップ607を実行する。説明する必要がある点としては、もし該第1共通集合と該第2共通集合のうち、空集合でない少なくとも一項が存在し、且つ該ターゲットトランザクションが該並列トランザクションと競合しなければ、ノード機器は以下のステップ608を実行する。

【0251】

ステップ605：ノード機器はターゲットトランザクションと並列トランザクションからロールバックすべきトランザクションを決定する。

【0252】

本願の実施例において、上記ステップ604により、ノード機器はデータ異常が出現したと決定し、該ターゲットトランザクション、又は該並列トランザクションをロールバックする必要がある、この2つのトランザクションからロールバックすべきトランザクションを決定する。

10

20

30

40

50

## 【0253】

以下、本ステップ605に関する2種類の選択可能な実施形態を説明する。

## 【0254】

方式1：ノード機器はトランザクション優先度に応じて、該ターゲットトランザクションと該並列トランザクションのうちトランザクション優先度が最も低いトランザクションを該ロールバックすべきトランザクションとして決定する。

## 【0255】

本願の実施例において、トランザクション優先度は予め設定された優先度であり、データベースシステムは各種のトランザクションにトランザクション優先度を予め割り当てる。たとえば、トランザクションの処理期間、トランザクションの操作タイプ、及び作用した変数数量等のうちの少なくとも一種に応じて、各トランザクションのトランザクション優先度を決定するが、本願の実施例はトランザクション優先度の設定方式について限定しない。

10

## 【0256】

例示的には、以下、4種類のトランザクション優先度の設定方式を例として、本ステップ605の実施形態を説明する。

## 【0257】

第1種としては、もし該ターゲットトランザクションと該並列トランザクションのうちの現在のトランザクションの処理期間がその他のトランザクションの処理期間未満であれば、ノード機器は該現在のトランザクションをトランザクション優先度の最も低いトランザクションとして決定し、それにより、該ロールバックすべきトランザクションはすなわち該現在のトランザクションとなる。

20

## 【0258】

ここで、トランザクション優先度はトランザクションの処理期間に応じて決定される。処理期間が参照期間よりも大きいトランザクションはロングトランザクションであり、処理期間が参照期間以下のトランザクションはショートトランザクションとして決定される。ロングトランザクションのトランザクション優先度はショートトランザクションのトランザクション優先度よりも高い。ここでは、現在のトランザクションはターゲットトランザクションと並列トランザクションのうちのいずれか1つのトランザクションであり、その他のトランザクションはターゲットトランザクションと並列トランザクションのうちの現在のトランザクション以外のトランザクションである。

30

## 【0259】

例示的には、ターゲットトランザクションにとっては、並列トランザクションの処理期間を参照期間とし、もしターゲットトランザクションの処理期間が該並列トランザクション未満であれば、ノード機器は該ターゲットトランザクションをロールバックすべきトランザクションとして決定し、そうでなければ、並列トランザクションをロールバックすべきトランザクションとして決定する。

## 【0260】

第2種類としては、もし該ターゲットトランザクションと該並列トランザクションのうちのいずれか1つのトランザクションが読み取りトランザクションであれば、ノード機器は該読み取りトランザクションをトランザクション優先度の最も低いトランザクションとして決定し、それにより、ロールバックすべきトランザクションはすなわち該読み取りトランザクションである。

40

## 【0261】

ここで、トランザクション優先度はトランザクションの操作タイプに応じて決定され、書き込みトランザクションのトランザクション優先度は読み取りトランザクションのトランザクション優先度よりも高い。ここでは、トランザクションの操作タイプは書き込みトランザクションタイプ、及び読み取りトランザクションタイプを含み、書き込みトランザクションタイプのトランザクションは書き込みトランザクションであり、読み取りタイプのトランザクションは読み取りトランザクションである。

50

## 【0262】

例示的には、もしターゲットトランザクションが書き込みトランザクションであり、並列トランザクションが読み取りトランザクションであれば、ノード機器は該ターゲットトランザクションをロールバックすべきトランザクションとして決定する。

## 【0263】

第3種類としては、もし該ターゲットトランザクションと該並列トランザクションのうちの現在のトランザクションが書き込みトランザクションであり、且つ該書き込みトランザクションの処理期間が参照期間未満であれば、ノード機器は該書き込みトランザクションをトランザクション優先度の最も低いトランザクションとして決定し、それにより、ロールバックすべきトランザクションはすなわち該現在のトランザクションとなる。

10

## 【0264】

ここで、トランザクション優先度はトランザクションの処理期間、及び操作タイプを組み合わせて決定される。すなわち、書き込みトランザクションはショートトランザクションであるときに、そのトランザクション優先度がロング読み取りトランザクションよりも低い。例示的には、もし該ターゲットトランザクションが書き込みトランザクションであり、且つ該ターゲットトランザクションの処理期間が参照期間未満であれば、ノード機器は該ターゲットトランザクションをロールバックすべきトランザクションとして決定する。

## 【0265】

説明する必要がある点としては、いくつかの実施例において、トランザクション優先度の設定方式はさらにその他の形式を有することができる。たとえば、ショートトランザクションのトランザクション優先度はロングトランザクションのトランザクション優先度よりも高く、また、たとえば、読み取りトランザクションのトランザクション優先度は書き込みトランザクションのトランザクション優先度よりも高い等であるが、本願の実施例はこれについて限定しない。

20

## 【0266】

理解することができるように、このようなトランザクション優先度に基づいてロールバックすべきトランザクションを決定する方式により、トランザクションを適切にロールバックすることを可能にし、重要なトランザクションが優先的に処理されることを保証し、それにより計算リソースの利用率を向上させ、トランザクション処理の効率を向上させる。

## 【0267】

第4種類としては、もし該ターゲットトランザクションと該並列トランザクションのうちの現在のトランザクションが作用した変数数量がその他のトランザクションが作用した変数数量よりも大きければ、ノード機器は該現在のトランザクションをトランザクション優先度の最も低いトランザクションとして決定する。それにより、ロールバックすべきトランザクションはすなわち該現在のトランザクションとなる。

30

## 【0268】

本願の実施例において、ノード機器はターゲットトランザクションと並列トランザクションのターゲット競合閉路グラフをトラバーサルし（ターゲット競合閉路グラフについての紹介は後のステップ607において紹介されるようになり、ここで再度繰り返し記述しない）、現在のトランザクションが作用した変数数量が最も多いときに、該現在のトランザクションをロールバックすべきトランザクションとして決定する。

40

## 【0269】

このような選択可能な実施形態により、関する変数数量が比較的多いトランザクションをロールバックすることで、後続のトランザクション処理過程においてその他のトランザクションと競合する可能性を減少させることを可能にし、それによりトランザクション処理の競合率を減少させ、トランザクション処理の効率を向上させる。

## 【0270】

方式2：ノード機器はターゲットトランザクションと並列トランザクションからランダムに選択したトランザクションを、ロールバックすべきトランザクションとして決定する。

## 【0271】

50

ステップ606：ノード機器は該ロールバックすべきトランザクションをロールバックする。

【0272】

説明する必要がある点としては、上記ステップ601～ステップ606により、ノード機器はターゲットトランザクションと並列トランザクションとが競合を構成するか否かを決定し、データ異常が存在する場合のみに、ロールバックすべきトランザクションを決定する。

【0273】

理解することができるように、ノード機器はロールバックすべきトランザクションを決定するときに、ターゲットトランザクションを直接ロールバックするのではなく、所定の戦略に応じて決定する。このように、偽ロールバック率を効果的に低減させることを可能にし、それにより計算リソースの利用率を向上させ、トランザクション処理の効率を向上させる。

【0274】

ステップ607：ノード機器はターゲット競合閉路グラフにおけるロールバックすべきトランザクションが作用した変数、及び対応する読み取り書き込み操作を削除する。ここで、ターゲット競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられ、ターゲット競合閉路グラフにおける辺はトランザクション間の関連関係、及び変数間のバージョン変化を表すことに用いられる。

【0275】

本願の実施例において、ターゲット競合閉路グラフは一種の細部競合閉路グラフ（Detail Conflict Cycle Graph、DCCG）であって、CCGグラフ、及びCCSGグラフと等価である一種の競合閉路グラフであり、本願の実施例におけるデータ状態整合性モデルの実現基礎でもある。上記ステップ606により、ロールバックすべきトランザクションをロールバックした後に、ノード機器はターゲット競合閉路グラフにおけるロールバックすべきトランザクションが作用した変数、及び対応する読み取り書き込み操作を削除して、ターゲット競合閉路グラフの構造を維持する。

【0276】

以下、ターゲット競合閉路グラフの取得方式を詳細に論述する。下記ステップはステップ1～ステップ3を含み、以下、各ステップをそれぞれ説明する。

【0277】

ステップ1：該ターゲットトランザクションと該並列トランザクションとの間の第1競合閉路グラフを取得する。該第1競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子を表すことに用いられ、該第1競合閉路グラフにおける辺はトランザクション間の関連関係、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられる。

【0278】

ここで、第1競合閉路グラフは該ターゲットトランザクションと該並列トランザクションとの間のCCGグラフである。第1競合閉路グラフにおける頂点、及び辺の意味についてCCGグラフの頂点、及び辺についての上記紹介を参照すればよく、ここでは再度繰り返し記述しない。

【0279】

ステップ2：該第1競合閉路グラフを変換して、第2競合閉路グラフを得る。該第2競合閉路グラフにおける頂点は複数の異なるバージョンの変数を表すことに用いられ、該第2競合閉路グラフにおける辺は変数間のバージョン変化、及び同一トランザクションが作用した変数を表すことに用いられる。

【0280】

ここで、第2競合閉路グラフは該ターゲットトランザクションと該並列トランザクションとの間のCCSGグラフである。第2競合閉路グラフにおける頂点、及び辺の意味につ

10

20

30

40

50

いて、CCSGグラフの頂点、及び辺についての上記紹介を参照すればよく、ここでは再度繰り返し記述しない。CCGグラフ、及びCCSGグラフについての上記紹介に基づいて明らかなように、CCGグラフはCCSGグラフと等価であり、以下、CCGグラフからCCSGグラフへの等価変換規則を基礎として、本ステップ2の実施形態を簡単に説明する。以下のステップA、及びステップBを含み、以下、各ステップをそれぞれ説明する。

【0281】

ステップA：該第1競合閉路グラフの辺を分割して、該複数の異なるバージョンの変数を得る。

【0282】

ステップB：該複数の異なるバージョンの変数を頂点として、隣接するバージョンの同一変数に対応する頂点の間に辺を追加し、同一トランザクションが作用した変数に対応する頂点の間に辺を追加して、該第2競合閉路グラフを得る。

【0283】

ステップ3：該第2競合閉路グラフを変換して、該ターゲット競合閉路グラフを得る。

【0284】

ここで、CCGグラフ、及びCCSGグラフについての上記紹介に基づいて明らかなように、CCGグラフはCCSGグラフと等価であり、CCSGグラフはCCGグラフに等価変換することを可能にし、該ターゲット競合閉路グラフはCCSGグラフを変換することにより得られた新しいCCGグラフである。上記第1競合閉路グラフにおいて、該第1競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子を表すことに用いられる。一方、ターゲット競合閉路グラフ（すなわち新しいCCGグラフである）において、該ターゲット競合閉路グラフの頂点情報は第1競合閉路グラフにおける頂点情報に比べてより細分化され、トランザクションに対応するトランザクション識別子を含むだけでなく、トランザクションに対応する読み取り書き込み操作が作用した変数を含む。また、該ターゲット競合閉路グラフの頂点に対応する変数は競合閉路を構成する必要変数を残し、1つのトランザクションが操作した変数は非常に多い可能性があるが、すべての変数、及びそのバージョンはいずれも競合閉路グラフを構成する十分条件であるわけではない。

【0285】

以下、CCSGグラフからCCGグラフへの上記等価変換規則を基礎として、本ステップ3の具体的な実施形態を説明する。以下のステップC～ステップEを含み、以下、各ステップをそれぞれ説明する。

【0286】

ステップC：該第2競合閉路グラフにおける同一トランザクションが作用した変数に対応する頂点を合併し、且つ同一トランザクションに対応するトランザクション識別子を合併後の頂点として決定する。すなわち、ノード機器は合併された後の頂点を同一トランザクションに対応するトランザクション識別子と命名する。

【0287】

ステップD：合併後の第2競合閉路グラフの隣接する頂点の間に辺を追加して、第3競合閉路グラフを得る。

【0288】

ステップE：もし該第3競合閉路グラフの任意の2つの隣接する頂点中に同じバージョンの変数が存在すれば、該任意の2つの隣接する頂点を合併し、且つ合併後の頂点と対応する該同じバージョンの変数を消去し、該ターゲット競合閉路グラフを得る。もし該第3競合閉路グラフの任意の2つの隣接する頂点中に異なるバージョンの同一変数が存在すれば、該任意の2つの隣接する頂点を合併して該ターゲット競合閉路グラフを得る。

【0289】

ここで、該第3競合閉路グラフは、すなわちCCSGグラフに対して等価変換規則に応じて変換を行うことにより得られたCCGグラフであり、該第3競合閉路グラフにおいて、もし任意の2つの隣接する頂点中に同じバージョンの変数が存在すれば、該第3競合閉

10

20

30

40

50

路グラフは縮約規則に適用され、それによりノード機器はこの2つの頂点を合併し、且つ同じバージョンの変数を消去する。もし任意の2つの隣接する頂点中に異なるバージョンの同一変数が存在すれば、該第3競合閉路グラフは合併規則に適用されるが縮約規則に適用されず、それによりノード機器はこの2つの頂点を合併し、もし任意の2つの隣接する頂点中に同じ変数が存在しなければ、ノード機器は直接該第3競合閉路グラフをターゲット競合閉路グラフとする。

【0290】

以下、上記縮約規則、及び合併規則を紹介する。

【0291】

第1、縮約規則。

10

【0292】

1つの競合閉路グラフにおける任意の2つの隣接する頂点 $V_i$ 及び $V_{i+1}$ に対して、もし $V_i \cap V_{i+1} = \{x_m, y_n, \dots\} \neq [\text{数3}]$ であり、すなわち少なくとも1つの同じバージョンの変数が存在すれば、 $V_i$ 及び $V_{i+1}$ は1つの頂点として合併することができる。該合併された新しい頂点において、 $x_m$ 及び $y_n$ 等は除去することができ、2つの頂点中の互いに交差しない部分を残し、すなわち $V_i \cup V_{i+1} - V_i \cap V_{i+1}$ である。すなわち縮約後の新しい頂点は $V_i$ 及び $V_{i+1}$ という2つの集合の差集合を残す。

【0293】

【数3】

$\emptyset$

20

【0294】

ターゲット競合閉路グラフに対する上記紹介、及び該縮約規則に基づいて、以下の3つの推論を得ることを可能にし、以下、各推論をそれぞれ説明する。

【0295】

推論1：もし1つのスケジュール $s$ において、 $\exists d c c g \in D C C G$ であり、該 $d c c g$ グラフの任意の2つの隣接する頂点の共通集合が空集合であれば、該 $d c c g$ は最小 $D C C G$ グラフと呼称され、すなわち該 $d c c g$ グラフにおいて縮約規則を満たす頂点の縮約を完了した後に、得られた $D C C G$ グラフが最も簡潔である。

30

【0296】

推論2：1つの最小 $D C C G$ グラフは上記表2におけるあるデータ異常と等価である。

【0297】

推論3：もし1つのスケジュール $s$ において、任意の2つの並列トランザクション間に $W R$ 関係が存在すれば、この2つの並列トランザクションは縮約規則に適用することができる。

【0298】

第2、合併規則。

【0299】

1つの競合閉路グラフにおける任意の2つの隣接する頂点 $V_i$ 及び $V_{i+1}$ に対して、もし $V_i \cap V_{i+1} = [\text{数4}]$ であれば、この2つの頂点は縮約することができない。もし $V_i \cap V_{i+1} = \{x, y, \dots\} \neq [\text{数4}]$ であり、すなわち少なくとも1つの同じ変数が存在し、且つ縮約規則に適用されなければ、この2つの隣接する頂点は合併規則に適用される。つまり、この2つの隣接する頂点に対応するトランザクションを合併可能なトランザクションと呼称し、 $V_i$ 及び $V_{i+1}$ を1つの頂点として合併し、該合併された新しい頂点に対応するトランザクションの読み取りセットは $V_i$ 及び $V_{i+1}$ に対応する読み取りセットの和集合であり、該合併された新しい頂点に対応するトランザクションの書き込みセットは $V_i$ 及び $V_{i+1}$ に対応する書き込みセットの和集合である。

40

【0300】

【数4】

50

∅

### 【0301】

また、ターゲット競合閉路グラフに対する上記紹介、及び該合併規則に基づいて以下の推論を得ることを可能にする。1つのスケジュール  $s$  の  $dccg$  グラフで表されたデータ異常は、合併可能なトランザクションを合併した後に、元の  $dccg$  グラフと等価なデータ異常に影響を与えない。

### 【0302】

以上は、ターゲット競合閉路グラフの取得方式、及び関連規則を紹介であり、以下、表4、及び図10を参照して、例示を組み合わせ、ターゲット競合閉路グラフの取得方式に対して例を挙げて説明する。ここで、図10は本願の実施例により提供されたターゲット競合閉路グラフを取得する模式図である。1つのスケジュール  $s$  は3つのトランザクション  $T_1$ 、 $T_2$ 、及び  $T_3$  を含み、その実行シーケンスは表4に示すとおりである。

### 【0303】

【表4】

表4

| 時点    | $T_1$       | $T_2$       | $T_3$       |
|-------|-------------|-------------|-------------|
| $t_0$ | $R_1 [x_0]$ |             |             |
| $t_1$ |             | $W_2 [x_1]$ |             |
| $t_2$ |             | $W_2 [y_1]$ |             |
| $t_3$ |             |             | $R_3 [y_1]$ |
| $t_4$ |             |             | $W_3 [z_1]$ |
| $t_5$ | $W_1 [z_2]$ |             |             |

### 【0304】

図10における(a)に示すように、該スケジュール  $s$  の  $CCG$  グラフ（すなわち第1競合閉路グラフ）における辺はトランザクション間の関連関係、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられ、 $E_1 = (T_1 \rightarrow T_2, R_1 [x_0] W_2 [x_1])$ 、 $E_2 = (T_2 \rightarrow T_3, W_2 [y_1] R_3 [y_1])$ 、及び  $E_3 = (T_3 \rightarrow T_1, W_3 [z_1] W_1 [z_2])$  という3つの辺を含み、3つの変数  $x$ 、 $y$ 、及び  $z$  に対応する。

### 【0305】

図10における(b)、及び(c)に示すように、該  $CCG$  グラフにおいて3つの変数  $x$ 、 $y$ 、及び  $z$  を含み、それぞれ辺  $E_1$ 、 $E_2$ 、及び  $E_3$  にあり、変数を頂点として、3つの辺を分割する。たとえば、 $E_1$  は  $R_1 [x_0] W_2 [x_1]$  として表され、該辺を分割して、 $E_{11} = R_1 [x_0]$ 、及び  $E_{12} = W_2 [x_1]$  を得て、 $E_2$  は  $W_2 [y_1] R_3 [y_1]$  として表され、該辺を分割して、 $E_{21} = W_2 [y_1]$ 、及び  $E_{22} = R_3 [y_1]$  を得て、 $E_3$  は  $W_3 [z_1] W_1 [z_2]$  として表され、該辺を分割して、 $E_{31} = W_3 [z_1]$ 、及び  $E_{32} = W_1 [z_2]$  を得る。これにより、 $CCG$  グラフにおける頂点を削除する。

### 【0306】

図10における(d)に示すように、上記辺分割過程を経た後に、変数  $x$  の2つのバージョン  $x_0$ 、及び  $x_1$ 、変数  $y$  の1つのバージョン  $y_1$ 、及び変数  $z$  の2つのバージョン  $z_1$ 、及び  $z_2$  を得る。これらの変数のバージョンを  $CCSG$  グラフの頂点として、同一変数の複数のバージョンは縦方向に並んでおり、同一トランザクションに作用された異なる変数のバージョン間は横方向に並んでいる（理解すべき点としては、図10における(d)において変数  $z$  の初期バージョン  $z_0$  をさらに含み、このようにしてこそ辺  $E_{31}$  に対応することができる）。ここでは、隣接するバージョンの同一変数に対応する頂点の間に辺

を追加し、同一トランザクションが作用した変数に対応する頂点の間に辺を追加する。また、CCSGグラフに対する上記記述に基づいて明らかなように、CCSGグラフは有向閉路で構成された1つのグラフである。この場合、辺 $W_3[z_1]$ 、 $W_2[y_1]$ 、及び変数バージョン $z_0$ 、及び $y_0$ は有向閉路の構成に対して作用がなく、従って、この2本の辺、及びこの2つの変数バージョンを消去して、図10における(e)に示されたCCSGグラフを得る。

【0307】

図10における(e)に示すように、該CCSGグラフ(すなわち第2競合閉路グラフ)における頂点は複数の異なるバージョンの変数を表すことに用いられ、該第2競合閉路グラフにおける辺は変数間のバージョン変化、及び同一トランザクションが作用した変数を表すことに用いられる。例示的には、頂点 $V = \{x_0, x_1, y_1, z_1, z_2\}$ であり、 $E = \{W_2[x_1], R_2[x_1] R_2[y_1], R_3[y_1] R_3[z_1], W_1[z_2], R_1[x_2] R_1[x_0]\}$ であり、ここで、 $E_T = \{R_2[x_1] R_2[y_1], R_3[y_1] R_3[z_1], R_1[x_2] R_1[x_0]\}$ であり、 $E_S = \{W_2[x_1], W_1[z_2]\}$ である。

【0308】

図10における(f)に示すように、該図10における(f)は図10における(e)に示されたCCSGグラフを変換することにより得られた新しいCCGグラフ(すなわち第3競合閉路グラフ)である。変換過程は、ノード機器がCCSGグラフにおける同一トランザクションに関する変数バージョンを同一頂点内部に合併して該図10における(f)を得ることを含む。図10における(f)に示すように、各頂点はトランザクションを頂点とするように改めて変更され、トランザクション間の関連関係を示している。説明する必要がある点としては、いくつかの実施例において、該グラフはさらに1つの無向閉路グラフであってもよい。

【0309】

図10における(g)に示すように、頂点 $V_2$ 、及び $V_3$ は上記縮約規則を満たし、すなわちこの2つの頂点中にはいずれも同じバージョンの変数 $y_1$ を含む。従って、頂点 $V_2$ 、及び $V_3$ を合併して、合併後の頂点における変数 $y_1$ を消去して図10における(h)に示されたDCCGグラフ(すなわちターゲット競合閉路グラフ)を得る。

【0310】

以上、例を挙げる方式によりターゲット競合閉路グラフの取得方式を紹介したため、並列トランザクション間の有向閉路グラフを決定する過程として理解することもできる。本ステップ607において、ノード機器はロールバックすべきトランザクションを決定した後に、該ターゲット競合閉路グラフの構造を維持することで、後続で次のサイクルを実行しやすくなり、すなわち、少なくとも1つの候補並列トランザクションのうちのその他の候補並列トランザクションに対して整合性検査を行う。

【0311】

以下、ステップ608により、ターゲットトランザクションが並列トランザクションと競合しない場合を説明する。

【0312】

ステップ608：ノード機器はターゲットトランザクションと並列トランザクションを合併して合併後のターゲットトランザクションを得て、上記ステップ603を再び実行し、タイプスクリプトTSが空になるまで続ける。

【0313】

本願の実施例において、上記ステップ604により、第1共通集合と第2共通集合のうち空集合でない少なくとも一項目が存在し、且つ該ターゲットトランザクションが該並列トランザクションと競合せず、該ターゲットトランザクションと該並列トランザクションとの間にターゲット競合閉路グラフを構成しないとして理解することもできる。この場合、ターゲットトランザクションと並列トランザクションにデータ異常が存在せず、ノード機器はこの2つのトランザクションを合併して、合併後のターゲットトランザクションを得

10

20

30

40

50

て、タイプスクリプト T S から候補並列トランザクションを継続的に選択し、タイプスクリプト T S が空になるまで続ける。

【0314】

以下、ターゲットトランザクションが T であり、並列トランザクションが T L であることを例として、本ステップ 608 の実施形態を説明する。

【0315】

ノード機器は該並列トランザクション T L の読み取りセットをターゲットトランザクション T の読み取りセットに組み込み、該並列トランザクション T L の書き込みセットをターゲットトランザクション T の書き込みセットに組み込むことで、トランザクション合併を実現して合併後の新しいトランザクション T - new を得る。また、もし並列トランザクション T L がコミットされていなければ、T - new のターゲットパラメータ (no\_\_committed) に対して増分処理を行い、すなわち T - new . no\_\_committed++ (1 増やす) にし、合併により得られた新しいトランザクションにおいて、ある既にコミットされたトランザクションの読み取り書き込み集合の成分を持つことを表す。これに基づいて、ノード機器は T - new を新しいターゲットトランザクション T ' (すなわち T ' = T - new) として決定し、タイプスクリプト T S から 2 番目の候補並列トランザクションを取り出して T L ' として決定し、ステップ 603 (新しい T - new と T L ' との間が競合するか否かを判断する) を実行する。

【0316】

ステップ 609 : もしタイプスクリプト T S における最後の 1 つの候補並列トランザクションと該ターゲットトランザクションとの間にデータ異常が存在しなければ、ノード機器は該ターゲットトランザクションをコミットする。

【0317】

本願の実施例において、もしタイプスクリプト T S における最後の 1 つの候補並列トランザクションと該ターゲットトランザクションとの間にデータ異常が存在しなければ、ターゲットトランザクションがコミットすることができることが表明される。ここでは、もし隔離レベルパラメータ = S であり、すなわち隔離レベルが直列化可能レベルであれば、このときに、直列化可能レベルを満たすことができ、そうでなければ、上記 [表 3] により設定された隔離レベルを満たす。

【0318】

上記すべての選択可能な技術的解決手段は、任意の組み合わせを採用して本願の実施例を形成することができ、ここで再度 1 つずつ記述はしない。

【0319】

図 11 は本願の実施例に従って提供されたトランザクション処理装置の構造模式図である。該トランザクション処理装置は上記トランザクション処理方法が実行される時のステップを実行することに用いられ、図 11 に参照されるように、該トランザクション処理装置 1100 は、第 1 決定モジュール 1101 と、第 2 決定モジュール 1102 とを含む。

【0320】

第 1 決定モジュール 1101 は、ターゲットトランザクションの並列トランザクションを決定するように配置され、上記並列トランザクションと上記ターゲットトランザクションとの間には同一変数に作用する読み取り書き込み操作を含み、上記ターゲットトランザクションはコミットすべきトランザクションであり、

第 2 決定モジュール 1102 は、上記ターゲットトランザクションの読み取りセットと上記並列トランザクションの書き込みセットとの間の第 1 共通集合を取得し、且つ上記ターゲットトランザクションの書き込みセットと上記並列トランザクションの読み取りセットとの間の第 2 共通集合を取得し、上記第 1 共通集合と上記第 2 共通集合のうちの少なくとも一項が空でないデータセットであり、且つ上記ターゲットトランザクションが上記並列トランザクションと競合するときに、上記ターゲットトランザクションと上記並列トランザクションが作用した同一変数のバージョン変更回数、及びターゲット変数リストに基づいて、データ異常タイプを決定するように配置され、上記ターゲット変数リストは上記

10

20

30

40

50

ターゲットトランザクションと上記並列トランザクションが作用した同一変数を含む。

【0321】

本願の実施例において、該トランザクション処理装置1100は第3決定モジュールとロールバックモジュールとをさらに含み、第3決定モジュールは、該ターゲットトランザクション、及び該並列トランザクションからロールバックすべきトランザクションを決定するように配置され、ロールバックモジュールは、該ロールバックすべきトランザクションをロールバックするように配置される。

【0322】

本願の実施例において、第3決定モジュールは、トランザクション優先度に基づいて、上記ターゲットトランザクションと上記並列トランザクションのうちトランザクション優先度の最も低いトランザクションを上記ロールバックすべきトランザクションとして決定するように配置される。

10

【0323】

本願の実施例において、該第3決定モジュールは、上記ターゲットトランザクションと上記並列トランザクションにおいて、現在のトランザクションの処理期間がその他のトランザクションの処理期間未満であるときに、上記現在のトランザクションを上記トランザクション優先度の最も低いトランザクションとして決定し、上記現在のトランザクションは上記ターゲットトランザクションと上記並列トランザクションのうちのいずれか1つのトランザクションであり、上記その他のトランザクションは上記ターゲットトランザクションと上記並列トランザクションのうちの上記現在のトランザクション以外のトランザクションであり、上記ターゲットトランザクションと上記並列トランザクションにおいて、上記現在のトランザクションが読み取りトランザクションであるときに、上記現在のトランザクションを上記トランザクション優先度の最も低いトランザクションとして決定し、上記ターゲットトランザクションと上記並列トランザクションにおいて、上記現在のトランザクションが書き込みトランザクションであり、且つ上記現在のトランザクションの処理期間が参照期間未満であるときに、上記現在のトランザクションを上記トランザクション優先度の最も低いトランザクションとして決定し、上記ターゲットトランザクションと上記並列トランザクションにおいて、上記現在のトランザクションが作用した変数数量が上記その他のトランザクションが作用した変数数量よりも大きいときに、上記現在のトランザクションを上記トランザクション優先度の最も低いトランザクションとして決定するように配置される。

20

30

【0324】

本願の実施例において、該トランザクション処理装置1100は削除モジュールをさらに含み、ターゲット競合閉路グラフにおける該ロールバックすべきトランザクションが作用した変数、及び対応する読み取り書き込み操作を削除するように配置される。ここで、該ターゲット競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用いられ、該ターゲット競合閉路グラフにおける辺はトランザクション間の関連関係、及び変数間のバージョン変化を表すことに用いられる。

【0325】

40

本願の実施例において、該トランザクション処理装置1100は取得モジュールをさらに含み、該ターゲットトランザクションと該並列トランザクションとの間の該ターゲット競合閉路グラフを取得するように配置される。

【0326】

本願の実施例において、該取得モジュールはサブ取得モジュールと、第1変換モジュールと、第2変換モジュールとを含み、取得モジュールは、該ターゲットトランザクションと該並列トランザクションとの間の第1競合閉路グラフを取得するように配置され、該第1競合閉路グラフにおける頂点はトランザクションに対応するトランザクション識別子を表すことに用いられ、該第1競合閉路グラフにおける辺はトランザクション間の関連関係、及びトランザクションに対応する読み取り書き込み操作が作用した変数を表すことに用

50

いられ、第1変換モジュールは、該第1競合閉路グラフを変換して、第2競合閉路グラフを得るように配置され、該第2競合閉路グラフにおける頂点は複数の異なるバージョンの変数を表すことに用いられ、該第2競合閉路グラフにおける辺は変数間のバージョン変化、及び同ートランザクションが作用した変数を表すことに用いられ、第2変換モジュールは、該第2競合閉路グラフを変換して、該ターゲット競合閉路グラフを得るように配置される。

#### 【0327】

本願の実施例において、該第1変換モジュールは、該第1競合閉路グラフの辺を分割して、該複数の異なるバージョンの変数を得て、該複数の異なるバージョンの変数を頂点とし、且つ隣接するバージョンの同一変数に対応する頂点の間に辺を追加し、同ートランザクションが作用した変数に対応する頂点の間に辺を追加して、該第2競合閉路グラフを得るように配置される。

#### 【0328】

本願の実施例において、該第2変換モジュールは、上記第2競合閉路グラフにおける同ートランザクションが作用した変数に対応する頂点を合併し、且つ同ートランザクションに対応するトランザクション識別子を合併後の頂点として決定し、合併後の上記第2競合閉路グラフの隣接する頂点の間に辺を追加して、第3競合閉路グラフを得て、上記第3競合閉路グラフの2つの隣接する頂点中に同一バージョンの変数を含むときに、上記2つの隣接する頂点を合併し、且つ合併後の頂点における同一バージョンの変数を消去して、上記ターゲット競合閉路グラフを得て、上記第3競合閉路グラフの2つの隣接する頂点中に異なるバージョンの同一変数を含むときに、上記2つの隣接する頂点を合併し、上記ターゲット競合閉路グラフを得るように配置される。

#### 【0329】

本願の実施例において、該トランザクション処理装置1100は、第1処理モジュールと、第2処理モジュールと、第3処理モジュールとをさらに含み、第1処理モジュールは、上記第1共通集合が上記空でないデータセットであるときに、上記ターゲットトランザクションと上記並列トランザクションとの間の有向辺の数量に対して増分処理を行い、且つ上記第1共通集合における変数を上記ターゲット変数リスト中に追加し、上記第1共通集合中にバージョンが異なる変数を含むときに、上記バージョン変更回数に対して増分処理を行うように配置される。上記有向辺の数量は上記ターゲットトランザクションと上記並列トランザクションとの間が競合するか否かを指示することに用いられる。第2処理モジュールは、上記第2共通集合が上記空でないデータセットであるときに、上記有向辺の数量に対して増分処理を行い、且つ上記第2共通集合における変数を上記ターゲット変数リスト中に追加し、上記第2共通集合中にバージョンが異なる変数を含むときに、上記バージョン変更回数に対して増分処理を行うように配置される。第3処理モジュールは、第3共通集合が上記空でないデータセットであるときに、上記並列トランザクションのコミット状態に基づいて、上記データ異常タイプを決定するように配置される。上記第3共通集合は上記ターゲットトランザクションの書き込みセットと上記並列トランザクションの書き込みセットとの間の共通集合である。

#### 【0330】

本願の実施例において、該第2決定モジュール1102は、第1サブ決定モジュールと、第2サブ決定モジュールと、第3サブ決定モジュールとを含み、第1サブ決定モジュールは、上記ターゲットトランザクションと上記並列トランザクションにおいて同一変数に作用するターゲット読み取り書き込み操作に基づいて、所属カテゴリを決定するように配置される。第2サブ決定モジュールは、上記ターゲットトランザクションと上記並列トランザクションにおいて同一変数に作用するターゲット読み取り書き込み操作に基づいて、所属カテゴリにおけるサブカテゴリを決定するように配置される。第3サブ決定モジュールは、上記所属カテゴリにおける上記サブカテゴリ、上記バージョン変更回数、及び上記ターゲット読み取り書き込み操作のうちの少なくとも一種に基づいて、上記データ異常タイプを決定するように配置される。

## 【0331】

本願の実施例において、該第1サブ決定モジュールはさらに、上記ターゲットトランザクションと上記並列トランザクションにおける上記ターゲット読み取り書き込み操作が書き込み異常条件を満たすときに、上記所属カテゴリが書き込み異常カテゴリであると決定する。上記書き込み異常条件を満たすとは、上記ターゲット読み取り書き込み操作が異なるバージョンの同一変数に作用する第1指定数量の書き込み操作を含むことを指し、上記ターゲット読み取り書き込み操作が上記書き込み異常条件を満たさず、且つ読み取り異常条件を満たすときに、上記所属カテゴリが読み取り異常カテゴリであると決定する。上記読み取り異常条件を満たすとは、上記ターゲット読み取り書き込み操作が同一バージョンの同一変数に作用する読み取り操作と書き込み操作を含むことを指し、上記ターゲット読み取り書き込み操作が上記書き込み異常条件と上記読み取り異常条件を満たさず、且つ交差異常条件を満たすときに、上記所属カテゴリが交差異常カテゴリであると決定するように配置される。上記交差異常条件を満たすとは、上記ターゲット読み取り書き込み操作が異なるバージョンの同一変数に作用する読み取り操作と書き込み操作を含むことを指す。

10

## 【0332】

本願の実施例において、該第2サブ決定モジュールはさらに、上記ターゲット変数リストにおける変数数量が第2指定数量であるときに、上記所属カテゴリにおける上記サブカテゴリがシングルメタデータ異常であると決定し、上記ターゲット変数リストにおける変数数量が第3指定数量であるときに、上記所属カテゴリにおける上記サブカテゴリがダブルメタデータ異常であると決定し、上記ターゲット変数リストにおける変数数量が第4指定数量であるときに、上記所属カテゴリにおける上記サブカテゴリがマルチメタデータ異常であると決定するように配置される。ここで、上記第4指定数量は、上記第2指定数量、及び上記第3指定数量のいずれとも異なる。

20

## 【0333】

本願の実施例において、上記所属カテゴリが書き込み異常カテゴリであり、且つ上記サブカテゴリがシングルメタデータ異常であるときに、該第3サブ決定モジュールはさらに、上記バージョン変更回数が第2指定数量であるときに、上記書き込み異常カテゴリ、及び上記シングルメタデータ異常に基づいて、上記データ異常タイプが紛失自己更新異常であると決定し、上記バージョン変更回数が第3指定数量であり、且つ上記ターゲット読み取り書き込み操作が第5指定数量の書き込み操作であるときに、上記書き込み異常カテゴリ、及び上記シングルメタデータ異常に基づいて、上記データ異常タイプが全書き込み異常であると決定し、上記バージョン変更回数が上記第3指定数量であり、且つ上記ターゲット読み取り書き込み操作が第1指定数量の書き込み操作と第6指定数量の読み取り操作であるときに、上記書き込み異常カテゴリ、及び上記シングルメタデータ異常に基づいて、上記データ異常タイプが紛失更新異常であると決定するように配置される。

30

## 【0334】

本願の実施例において、上記所属カテゴリが書き込み異常カテゴリであり、且つ上記サブカテゴリがダブルメタデータ異常であり、上記バージョン変更回数が第2指定数量であるときに、第3サブ決定モジュールはさらに、上記ターゲット読み取り書き込み操作が第1条件と第2条件を満たすときに、上記書き込み異常カテゴリ、上記ダブルメタデータ異常、及び上記第2指定数量に基づいて、上記データ異常タイプが読み取り書き込み半順序異常であると決定する。上記第1条件を満たすとは、上記ターゲット読み取り書き込み操作が異なるバージョンの第1変数に作用する書き込み操作と読み取り操作を含むことを指し、上記第2条件を満たすとは、上記ターゲット読み取り書き込み操作が異なるバージョンの第2変数に作用する第1指定数量の書き込み操作を含むことを指し、上記ターゲット読み取り書き込み操作が第3条件と上記第2条件を満たすときに、上記書き込み異常カテゴリ、上記ダブルメタデータ異常、及び上記第2指定数量に基づいて、上記データ異常タイプがリピータブルライト半順序異常であると決定し、上記第3条件を満たすとは、上記ターゲット読み取り書き込み操作が同一バージョンの上記第1変数に作用する書き込み操作と読み取り操作を含むことを指し、上記ターゲット読み取り書き込み操作が第4条件と

40

50

上記第2条件を満たすときに、上記書き込み異常カテゴリ、上記ダブルメタデータ異常、及び上記第2指定数量に基づいて、上記データ異常タイプが全書き込み半順序異常であると決定するように配置され、上記第4条件を満たすとは、上記ターゲット読み取り書き込み操作が異なるバージョンの上記第1変数に作用する上記第1指定数量の書き込み操作を含むことを指す。

#### 【0335】

本願の実施例において、上記所属カテゴリが読み取り異常カテゴリであり、且つ上記サブカテゴリがシングルメタデータ異常であり、上記バージョン変更回数が第2指定数量であるときに、第3サブ決定モジュールはさらに、上記ターゲット読み取り書き込み操作が同一変数に作用する第1指定数量の読み取り操作、及び第6指定数量の書き込み操作を含む場合、上記読み取り異常カテゴリ、上記シングルメタデータ異常、及び上記第2指定数量に基づいて、上記データ異常タイプがノンリピータブルリード異常であると決定する。上記ターゲット読み取り書き込み操作が同一変数に作用する上記第1指定数量の書き込み操作と上記第6指定数量の読み取り操作を含む場合、上記読み取り異常カテゴリ、上記シングルメタデータ異常、及び上記第2指定数量に基づいて、上記データ異常タイプが中間読み取り異常であると決定するように配置される。

#### 【0336】

本願の実施例において、上記所属カテゴリが読み取り異常カテゴリであり、且つ上記サブカテゴリがダブルメタデータ異常であるときに、第3サブ決定モジュールはさらに、上記バージョン変更回数が第7指定数量であるときに、上記読み取り異常カテゴリと上記ダブルメタデータ異常に基づいて、上記データ異常タイプが書き込み読み取り半順序異常であると決定し、上記バージョン変更回数が第2指定数量であるときに、上記読み取り異常カテゴリと上記ダブルメタデータ異常に基づいて、上記データ異常タイプが読み取り半順序異常であると決定するように配置される。

#### 【0337】

本願の実施例において、第3サブ決定モジュールはさらに、上記所属カテゴリが交差異常カテゴリであり、且つ上記サブカテゴリがダブルメタデータ異常であり、及び上記バージョン変更回数が第2指定数量であるときに、上記データ異常タイプが書き込み半順序異常であると決定するように配置される。

#### 【0338】

本願の実施例において、該第3サブ決定モジュールはさらに、上記所属カテゴリが書き込み異常カテゴリであり、且つ上記サブカテゴリがマルチメタデータ異常であるときに、上記データ異常タイプが階段式書き込み異常であると決定し、上記所属カテゴリが読み取り異常カテゴリであり、且つ上記サブカテゴリが上記マルチメタデータ異常であるときに、上記データ異常タイプが階段式読み取り異常であると決定し、上記所属カテゴリが交差異常カテゴリであり、且つ上記サブカテゴリが上記マルチメタデータ異常であるときに、上記データ異常タイプが階段式交差異常であると決定するように配置される。

#### 【0339】

説明する必要がある点としては、上記実施例が提供したトランザクション処理装置は、トランザクションを処理するときについては、上記各機能モジュールの区分に対して例を挙げて説明したが、実際の応用においては、必要に従って上記機能を異なる機能モジュールに割り当てて完了させることができ、すなわち装置の内部構造を異なる機能モジュールに区分して、以上に記述したすべて、又は一部の機能を完了させ得る。また、本願の実施例により提供されたトランザクション処理装置はトランザクション処理方法と同一の思想に属するが、実現過程について詳しくは方法の実施例を参照すればよく、ここでは再度繰り返し記述しない。

#### 【0340】

本願の実施例はさらに電子機器（たとえば、コンピュータ機器）を提供しており、該電子機器はプロセッサと、メモリとを含み、該メモリは少なくとも1つのコンピュータプログラムを記憶することに用いられ、該少なくとも1セグメントのコンピュータプログラム

10

20

30

40

50

は該プロセッサによりロードされ且つ実行されて本願の実施例におけるトランザクション処理方法を実現する。

【0341】

いくつかの実施例において、本願の実施例に関するコンピュータプログラムは1つのコンピュータ機器上に配備されて実行されるか、又は1つの場所に位置する複数のコンピュータ機器上に実行されるか、又は、複数の場所に分散し且つ有線ネットワーク、又は無線ネットワークにより互いに接続される複数のコンピュータ機器上に実行されるようにしてもよく、複数の場所に分散し且つ有線ネットワーク、又は無線ネットワークにより互いに接続される複数のコンピュータ機器はブロックチェーンシステムを構成することができる。

【0342】

コンピュータ機器がサーバであることを例として、図12は本願の実施例に従って提供されたサーバの構造模式図である。該サーバ1200は配置、又は性能の異なりによって比較的大きな差異が発生する可能性があり、1つ、又は1つ以上のプロセッサ（Central Processing Units、CPU）1201、及び1つ、又は1つ以上のメモリ1202を含むことを可能にする。ここで、該メモリ1202において少なくとも1つのコンピュータプログラムが記憶されており、該少なくとも1つのコンピュータプログラムはプロセッサ1201によりロードされ且つ実行されて上記各々の方法実施例により提供されたトランザクション処理方法を実現する。もちろん、該サーバはさらに有線、又は無線ネットワークインタフェース、キーボード、及び入力出力インタフェース等の部材を含むことを可能にし、それにより入力出力を容易に行うことができ、該サーバはさらに機器の機能を実現することに用いられるその他の部材を含むことを可能にし、ここで再度繰り返し記述しない。

【0343】

本願の実施例はコンピュータ可読記憶媒体をさらに提供しており、該コンピュータ可読記憶媒体はコンピュータ機器に応用され、該コンピュータ可読記憶媒体において少なくとも1つのコンピュータプログラムが記憶されており、該少なくとも1つのコンピュータプログラムはプロセッサによりロードされ且つ実行されて本願の実施例のトランザクション処理方法を実現する。

【0344】

本願の実施例はコンピュータプログラム製品をさらに提供しており、コンピュータプログラム、又は指令を含み、該コンピュータプログラム、又は指令はコンピュータ可読記憶媒体において記憶されている。コンピュータ機器のプロセッサはコンピュータ可読記憶媒体から該コンピュータプログラム、又は指令を読み取り、プロセッサは該コンピュータプログラム、又は指令を実行して、該コンピュータ機器に本願の実施例により提供されたトランザクション処理方法を実行させる。

【0345】

当業者が理解することができるように、上記実施例のすべて、又は一部のステップの実現はハードウェアにより完了することができ、プログラムにより関連ハードウェアに指令を出して完了させることもでき、上記プログラムは一種のコンピュータ可読記憶媒体において記憶することができ、上記言及された記憶媒体は読み出し専用メモリ、磁気ディスク、又は光ディスク等であってもよい。

【0346】

理解することができるように、本願の実施例において、ユーザー情報等の関連データに関しては、本願の実施例は具体的な製品、又は技術に応用されるときに、ユーザーの許可、又は同意を取得する必要がある、且つ関連データの収集、使用、及び処理は関連国や地域の関連法律法規、及び標準を遵守する必要がある。

【0347】

上記は、本願の選択可能な実施例に過ぎず、本願を制限するためのものではなく、本願の精神、及び原則内で行ったいなか修正、等価置換、改良等は、いずれも本願の保護範囲内に含むべきである。

10

20

30

40

50

【符号の説明】

【 0 3 4 8 】

- 1 0 1    ゲートウェイサーバ
- 1 0 3    分散型記憶クラスター
- 1 0 4    分散型協調システム
- 1 1 0 0   トランザクション処理装置
- 1 1 0 1   第 1 決定モジュール
- 1 1 0 2   第 2 決定モジュール
- 1 2 0 0   サーバ
- 1 2 0 1   プロセッサ
- 1 2 0 2   メモリ

10

20

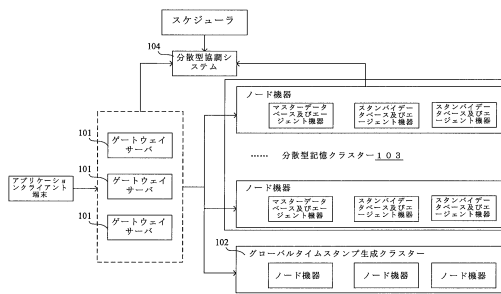
30

40

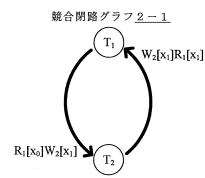
50

【図面】

【図 1】

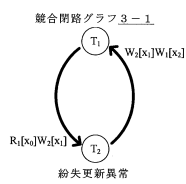


【図 2】

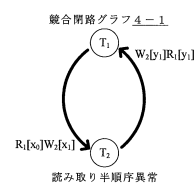


10

【図 3】



【図 4】



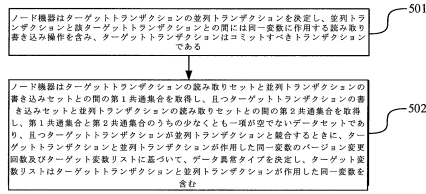
20

30

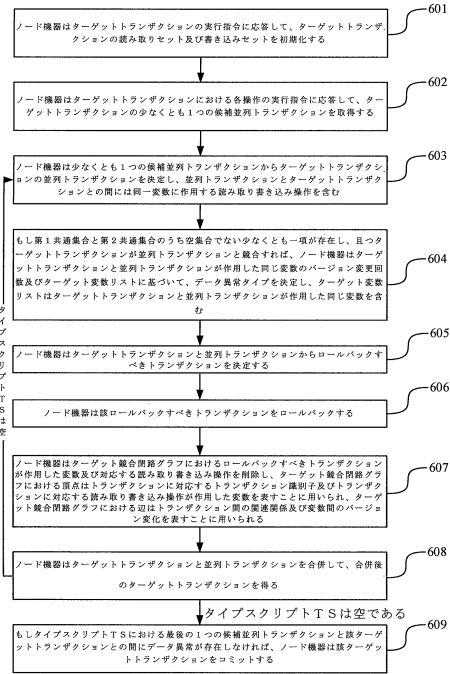
40

50

【図 5】



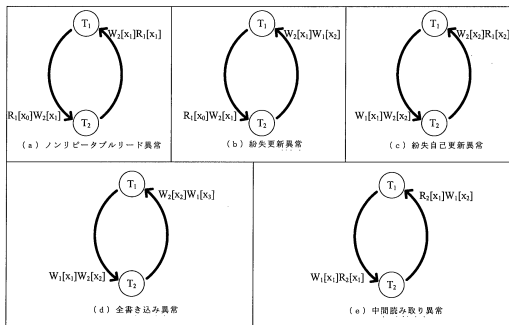
【図 6】



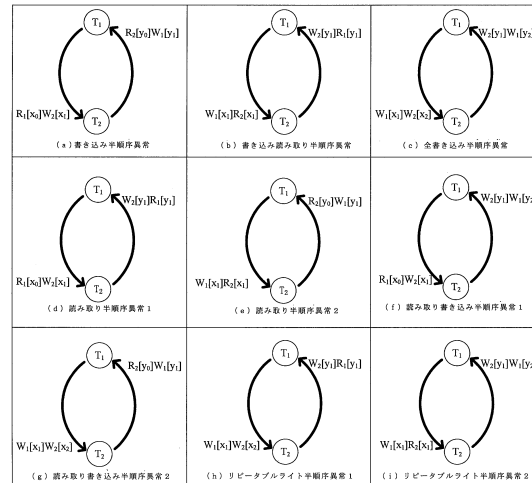
10

20

【図 7】



【図 8】

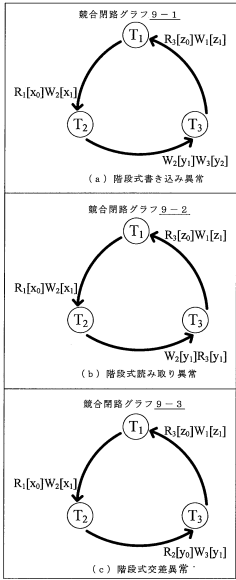


30

40

50

【図 9】



【図 10】

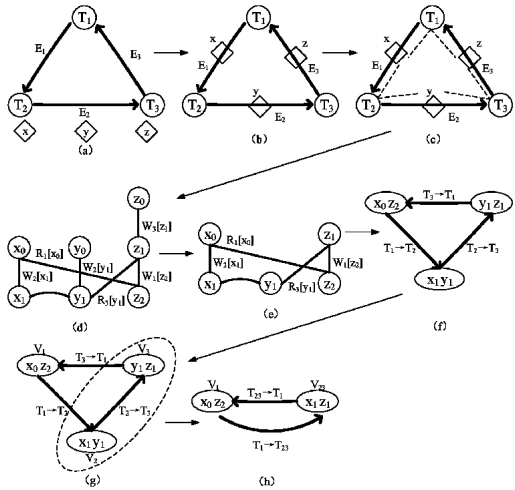
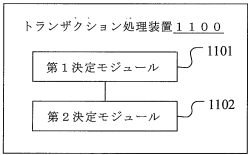


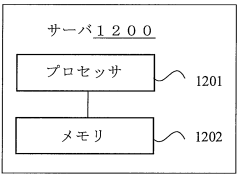
図 10

10

【図 11】



【図 12】



20

30

40

50

フロントページの続き

0 5 7 , C H I N A  
(74)代理人 100110364  
弁理士 実広 信哉  
(74)代理人 100150197  
弁理士 松尾 直樹  
(72)発明者 李 ▲海▼翔  
中華人民共和国 5 1 8 0 5 7 ▲広▼▲東▼省深▲セン▼市南山区高新区科技中一路 ▲騰▼▲訊▼  
大厦 3 5 ▲層▼  
審査官 松尾 真人  
(56)参考文献 中国特許出願公開第 1 1 1 7 0 8 6 1 5 ( C N , A )  
特開 2 0 2 0 - 2 0 4 9 7 5 ( J P , A )  
特開 2 0 1 5 - 1 1 8 4 3 1 ( J P , A )  
特開 2 0 2 1 - 0 5 1 7 4 3 ( J P , A )  
徐 海燕, Read Atomic 隔離性と並行実行の正当性の比較と検討, 電子情報通信  
学会技術研究報告 Vol. 1 1 7 No. 3 7 4 , 一般社団法人電子情報通信学会, 2017  
年12月15日, 第117巻, 第374号, pp.97~102  
(58)調査した分野 (Int.Cl., D B 名)  
G 0 6 F 1 6 / 0 0 - 1 6 / 9 5 8  
G 0 6 F 9 / 4 5 5 - 9 / 5 4