(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2016-48577
(P2016-48577A)
(43)公開日　平成28年4月7日(2016.4.7)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 12/00	(2006.01)	G06F	12/00	513D
G06F 9/50	(2006.01)	G06F	9/46	465E

審査請求 有　請求項の数 16　ＯＬ　（全 40 頁）

(21)出願番号　特願2015-235613 (P2015-235613)
(22)出願日　平成27年12月2日 (2015.12.2)
(62)分割の表示　特願2014-512271 (P2014-512271)の分割
原出願日　平成24年4月27日 (2012.4.27)

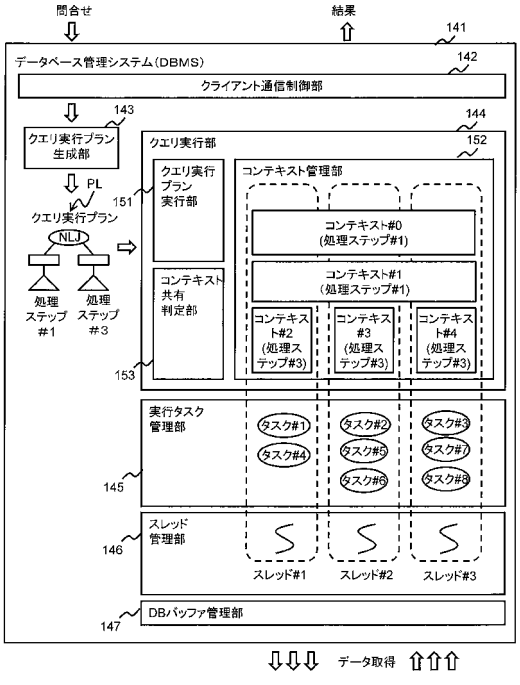
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(71)出願人　504137912
国立大学法人　東京大学
東京都文京区本郷七丁目３番１号
(74)代理人　110000279
特許業務法人ウィルフォート国際特許事務所
(72)発明者　清水　晃
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　徳田　晴介
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内

最終頁に続く

(54)【発明の名称】データベース管理システム、計算機、データベース管理方法

(57)【要約】　（修正有）
【課題】データベース管理システム（ＤＢＭＳ）が複数のプロセッサコアを使い、かつ、スレッドの管理オーバヘッドを小さくする。
【解決手段】ＤＢＭＳ１４１は、クエリを実行するために必要な１以上のデータベース（ＤＢ）オペレーションを表す情報を含むクエリ実行プランを生成し、そのクエリ実行プランに基づいて、クエリを実行する。そのクエリの実行において、ＤＢオペレーションを実行するためのタスクを動的に生成し、動的に生成されたタスクを実行する。プロセッサコアにより実行される複数のスレッドにおいてタスクを実行する。
【選択図】図１

Fig. 1



【特許請求の範囲】
【請求項１】
プロセッサコアを有する計算機により実現されデータベースを管理するデータベース管理システムであって、
前記データベースへのクエリを受け付けるクエリ受付部と、
前記受け付けたクエリを実行するために必要な処理ステップと前記処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成するクエリ実行プラン生成部と、
前記生成したクエリ実行プランに基づいて前記受け付けたクエリを実行し、前記受け付けたクエリの実行において、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行するクエリ実行部と
を有し、
前記クエリ実行部は、前記受け付けたクエリの実行において、プロセッサコアにより実行される複数のスレッドにおいてタスクを実行し、
前記プロセッサコアにより実行される一つのスレッドにおいて複数のタスクを実行し、
タスクを新たに生成する場合に、コンテキストを生成して、前記生成したコンテキストに基づいて前記生成されたタスクを実行し、
前記コンテキストは、前記新たに生成するタスクにおいて実行を開始する処理ステップが、前記クエリ実行プランが表す１以上の処理ステップのうちのいずれであるかを示す第１の情報と、前記第１の情報が示す処理ステップに要するデータのアクセス先に関する第２の情報と、前記新たに生成するタスクにより結果を生成するために必要なデータに関する第３の情報とを含む情報である
データベース管理システム。
【請求項２】
前記クエリ実行プラン生成部は、各前記処理ステップに関わるコンテキストを、複数のスレッド間で共有するか、複数のスレッド間で共有しないかを判定し、
前記クエリ実行部は、前記判定結果に基づいて前記コンテキストを管理する
請求項１に記載のデータベース管理システム。
【請求項３】
前記クエリ実行プラン生成部は、前記処理ステップの前記クエリ実行プランにおける他の処理ステップとの先行又は後続関係に基づいて、前記処理ステップに関わるコンテキストを複数のスレッド間で共有するか、又は共有しないかを判定する
請求項２に記載のデータベース管理システム。
【請求項４】
前記プロセッサコアにより実行されるスレッドに割当たったタスクにおいて、前記データベースに対して非同期Ｉ／Ｏによるデータ読み込み要求を発行し、
前記スレッドを実行するプロセッサコアは、前記タスクにおけるデータ読み込み要求発行後に、前記データ読み込み要求に対応するデータの読み込みが完了する前に、実行可能な他のタスクの実行を行い、
前記スレッドを実行するプロセッサコアは、前記タスクにおける前記データ読み込み要求に対応するデータの読み込みが完了した後に、前記タスクの実行を再開する
請求項２に記載のデータベース管理システム。
【請求項５】
前記処理ステップを実行するためのタスクを割り当て可能な前記スレッドは、前記プロセッサコアと同数であり、各スレッドを実行するプロセッサコアは、それぞれ別のプロセッサコアに設定されている
請求項２に記載のデータベース管理システム。
【請求項６】
前記クエリ実行プラン生成部は、前記クエリ実行プランに、並行して実行可能な処理ステップを含む複数の処理ブロックが含まれる場合に、前記処理ブロックの先頭の処理ステップに関わるコンテキストを、複数のスレッド間で共有すると判定し、前記処理ブロック

の他の処理ステップに関わるコンテキストを、スレッド間で共有しないと判定する
請求項２に記載のデータベース管理システム。
【請求項７】
前記クエリ実行プラン生成部は、前記クエリ実行プランに、並行して実行可能な処理ステップを含む複数の処理ブロックが含まれる場合に、前記処理ブロックにおける後続の処理ステップの数が所定数以上ある処理ステップに関わるコンテキストを、複数のスレッド間で共有すると判定し、後続の処理ステップの数が所定数未満の処理ステップに関わるコンテキストを、スレッド間で共有しないと判定する
請求項２に記載のデータベース管理システム。
【請求項８】
前記クエリ実行部は、一つのスレッドの利用可能なコンテキストの数が所定数を下回った場合に、他のスレッドの利用可能なコンテキストを、前記一つのスレッドに割当たったタスクで利用する
請求項２に記載のデータベース管理システム。
【請求項９】
前記クエリ実行部は、前記複数のスレッドに関わる実行状態が所定の状態となった場合に、スレッド間の利用可能なコンテキストの数の差が所定数より小さくなるように、一つのスレッドの利用可能なコンテキストを、他のスレッドの利用可能なコンテキストに変更する
請求項２に記載のデータベース管理システム。
【請求項１０】
前記複数のスレッドの実行状態が所定の状態とは、前記スレッド間の利用可能なコンテキストの数の差が所定数以上となった状態である
請求項９に記載のデータベース管理システム。
【請求項１１】
前記クエリ実行部は、データベース管理システムの実行状態に基づいて、コンテキストを複数のスレッドで共有とするか否かを判定し、
前記コンテキストを複数のスレッド間で共有すると判定されている場合には、前記コンテキストを複数のスレッド間で共有するようにし、前記コンテキストを複数のスレッド間で共有しないと判定されている場合には、前記コンテキストを一つのスレッドが利用するようにする
請求項１に記載のデータベース管理システム
【請求項１２】
前記データベース管理システムの実行状態とは、前記データベース管理システムにおいて現存するタスク数であり、
前記クエリ実行部は、現存する前記タスク数が所定数以下である場合には、前記コンテキストを複数のスレッドで共有とすると判定し、現存する前記タスク数が所定数以下でない場合には、前記コンテキストを共有しないと判定する
請求項１１に記載のデータベース管理システム。
【請求項１３】
前記データベース管理システムの実行状態とは、コンテキストに含まれる第３の情報であり、
前記クエリ実行部は、前記コンテキストに含まれる第３の情報のデータ量が所定量以下である場合には、前記コンテキストを複数のスレッドで共有とすると判定し、前記コンテキストに含まれる第３の情報のデータ量が所定量以下でない場合には、前記コンテキストを共有しないと判定する
請求項１２に記載のデータベース管理システム。
【請求項１４】
前記クエリ実行部は、前記コンテキストをいずれか一つのスレッドにより利用可能とし、

前記クエリ実行部は、前記スレッド間の利用可能なコンテキストの数の差が所定数以上となった場合に、前記スレッド間の利用可能なコンテキストの数の差が所定数より小さくなるように、一つのスレッドの利用可能なコンテキストを、他のスレッドの利用可能なコンテキストに変更する
請求項１に記載のデータベース管理システム。
【請求項１５】
記憶資源と、
前記記憶資源に接続され１以上のプロセッサコアを有する１以上のプロセッサを含んだ制御デバイスを有し、
前記制御デバイスが、
前記データベースへのクエリを受け付け、
前記受け付けたクエリを実行するために必要な処理ステップと前記処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成し、
前記生成したクエリ実行プランに基づいて前記受け付けたクエリを実行し、前記受け付けたクエリの実行において、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行し、
前記制御デバイスは、前記受け付けたクエリの実行において、プロセッサコアにより実行される複数のスレッドにおいてタスクを実行し、
前記プロセッサコアにより実行される一つのスレッドにおいて複数のタスクを実行し、
タスクを新たに生成する場合に、コンテキストを生成して、前記生成したコンテキストに基づいて前記生成されたタスクを実行し、
前記コンテキストは、前記新たに生成するタスクにおいて実行を開始する処理ステップが、前記クエリ実行プランが表す１以上の処理ステップのうちのいずれであるかを示す第１の情報と、前記第１の情報が示す処理ステップに要するデータのアクセス先に関する第２の情報と、前記新たに生成するタスクにより結果を生成するために必要なデータに関する第３の情報とを含む情報である
計算機。
【請求項１６】
データベースを管理するデータベース管理方法であって、
（ａ）前記データベースへのクエリを受け付け、
（ｂ）前記受け付けたクエリを実行するために必要な複数の１以上の処理ステップと、前記１以上の処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成し、
（ｃ）前記生成したクエリ実行プランに基づいて、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行することで、前記受け付けたクエリを実行し、
前記（ｃ）では、前記受け付けたクエリの実行において、プロセッサコアにより実行される複数のスレッドにおいてタスクを実行し、
前記プロセッサコアにより実行される一つのスレッドにおいて複数のタスクを実行し、
タスクを新たに生成する場合に、コンテキストを生成して、前記生成したコンテキストに基づいて前記生成されたタスクを実行し、
前記コンテキストは、前記新たに生成するタスクにおいて実行を開始する処理ステップが、前記クエリ実行プランが表す１以上の処理ステップのうちのいずれであるかを示す第１の情報と、前記第１の情報が示す処理ステップに要するデータのアクセス先に関する第２の情報と、前記新たに生成するタスクにより結果を生成するために必要なデータに関する第３の情報とを含む情報である
データベース管理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データ管理技術に関する。

【背景技術】
【０００２】
企業活動において、大量に生じる業務データの活用は不可欠になっている。そのため、大量の業務データを蓄積したデータベース（以下、「ＤＢ」）を解析処理するシステムが既に考案されている。
【０００３】
この解析処理において、データベース管理システム（以下、「ＤＢＭＳ」）は、クエリを受け付け、ＤＢを格納する記憶デバイスにデータ読出し要求を発行する。
【０００４】
１つのクエリの処理におけるデータ読出しの待ち時間の短縮化を図る技術として、特許文献１に開示の技術が知られている。特許文献１によれば、ＤＢＭＳは、クエリを実行するために必要な複数のデータベースオペレーション（ＤＢオペレーション、または、処理ステップと呼ぶ）を組合せたプラン（以下、クエリ実行プラン）を生成し、前記処理ステップを実行するタスクを動的に生成し、前記タスクを並行実行することでデータ読出し要求を多重化する。特許文献１によれば、タスクの実装としては、ＯＳが管理するプロセスやスレッド、または、アプリケーションやミドルウェアが実装する擬似プロセスや擬似スレッドなど任意の実行環境が利用できる。
【先行技術文献】
【特許文献】
【０００５】
【特許文献１】特開２００７－３４４１４号公報
【発明の概要】
【発明が解決しようとする課題】
【０００６】
ＤＢに蓄積したデータの数が数万ある場合、１つのクエリの処理に対して数万のタスクを生成する場合がある。このような場合にタスクをスレッド（またはプロセス）で実装するとスレッドが数万も生成される。複数のプロセッサコアを持つ計算機では、スレッドが任意のプロセッサコア上で実行するため、任意のプロセッサコアがスレッドの管理構造体を更新する。このため、スレッドの実行を管理するオーバヘッドが大きくなる。その結果、クエリの実行時間が伸びてしまう問題がある。
【０００７】
一方、擬似スレッド（または擬似プロセス）で実装すると、1つのスレッド（またはプロセス）で実装することになる。数万もの擬似スレッドを生成しても、擬似スレッドの管理構造体は１つのスレッドしか更新しないため、管理オーバヘッドは小さい。しかし、計算機が複数のプロセッサコアを持つ場合、スレッドが１つのためプロセッサコアを１つしか使わない。このため、処理ステップを実行するために必要となるプロセッサコアの処理が多い場合にプロセッサコアを１つしか使えないため、クエリの実行時間が伸びてしまう問題がある。
【０００８】
そこで、本発明の目的は、ＤＢＭＳが複数のプロセッサコアを使い、かつ、スレッドの管理オーバヘッドを小さくすることである。
【課題を解決するための手段】
【０００９】
ＤＢＭＳは、プロセッサコアを有する計算機により実現され、ＤＢを管理する。ＤＢＭＳは、ＤＢへのクエリを受け付けるクエリ受付部と、前記受け付けたクエリを実行するために必要な処理ステップと前記処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成するクエリ実行プラン生成部と、前記生成したクエリ実行プランに基づいて前記受け付けたクエリを実行し、前記受け付けたクエリの実行において、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行するクエリ実行部とを有する。

【００１０】
前記クエリ実行部は、前記受け付けたクエリの実行において、プロセッサコアにより実行される複数のスレッドにおいてタスクを実行し、前記プロセッサコアにより実行される一つのスレッドにおいて複数のタスクを実行する。例えば、クエリ実行部は、クエリの実行において、データベースオペレーションを実行するためのタスクを動的に生成し、動的に生成されたタスクを実行して良い。例えば、クエリ実行部は、クエリの実行において、（ａ）データベースオペレーションを実行するためのタスクを生成すること、（ｂ）生成されたタスクを実行することで、当該タスクに対応したデータベースオペレーションに必要なデータを読み出すためにデータベースへデータ読出し要求を発行すること、（ｃ）上記（ｂ）で実行されたタスクに対応したＮ番目のデータベースオペレーションの実行結果に基づき（Ｎ＋１）番目のデータベースオペレーションを実行する場合には、当該実行結果に基づくタスクを新たに生成すること（Ｎは１以上の整数）、及び、（ｄ）その新たに生成したタスクについて（ｂ）及び（ｃ）を行うこと、を行い、（ｂ）及び（ｄ）において、２以上の実行可能なタスクが存在する場合には、それら２以上のタスクのうちの少なくとも２つのタスクを並行して実行するようになっていて良い。
【００１１】
タスクを新たに生成する場合に、コンテキストを生成して、前記生成したコンテキストに基づいて前記生成されたタスクを実行する。前記コンテキストは、前記新たに生成するタスクにおいて実行を開始する処理ステップが、前記クエリ実行プランが表す１以上の処理ステップのうちのいずれであるかを示す第１の情報と、前記第１の情報が示す処理ステップに要するデータのアクセス先に関する第２の情報と、前記新たに生成するタスクにより結果を生成するために必要なデータに関する第３の情報とを含む情報である。
【発明の効果】
【００１２】
本発明によれば、一つのスレッドが複数のタスクを実行し、複数の前記スレッドでクエリを実行することにより、複数のプロセッサコアを使い、かつ、スレッドの管理オーバヘッドが小さくなる。その結果、クエリの実行時間が短縮できる。
【図面の簡単な説明】
【００１３】
【図１】図１は、実施例１に係るＤＢＭＳの概要を説明する図である。
【図２】図２は、実施例１に係るＤＢＭＳにおけるクエリの実行を説明する図である。
【図３】図３は、実施例１に係る計算機システムの構成図である。
【図４】図４は、実施例１に係るＤＢの表及び索引の定義を説明する図である。
【図５】図５は、実施例１に係るＤＢのＰａｒｔ表の一例を示す図である。
【図６】図６は、実施例１に係るＤＢのＬｉｎｅｉｔｅｍ表の一例を示す図である。
【図７】図７は、実施例１に係るＤＢにおけるＰａｒｔ索引及びＰａｒｔ表のデータ構造の一例を説明する図である。
【図８】図８は、実施例１に係るＤＢのクエリの一例を示す図である。
【図９】図９は、実施例１に係るクエリ実行プランの一例を示す図である。
【図１０】図１０は、実施例１に係るタスク管理情報のデータ構造の一例を示す図である。
【図１１】図１１は、実施例１に係るタスク実行状態情報の一例を示す図である。
【図１２】図１２は、実施例１に係る処理ステップ実行状態情報の第１の例を示す図である。
【図１３】図１３は、実施例１に係る処理ステップ実行状態情報の第２の例を示す図である。
【図１４】図１４は、実施例１に係る処理ステップ実行状態情報の第３の例を示す図である。
【図１５】図１５は、実施例１に係るコンテキスト管理情報のデータ構造の一例を説明する図である。

【図１６】図１６は、実施例１に係るスレッド＃１向け検索テーブルの一例を示す図である。
【図１７】図１７は、実施例１に係るスレッド＃２向け検索テーブルの一例を示す図である。
【図１８】図１８は、実施例１に係るスレッド＃３向け検索テーブルの一例を示す図である。
【図１９】図１９は、実施例１に係るコンテキストの一例を示す図である。
【図２０】図２０は、実施例１に係るクエリ受付時処理のフローチャートである。
【図２１】図２１は、実施例１に係るクエリ実行プラン生成処理のフローチャートである。
【図２２】図２２は、実施例１に係るスレッド間共有フラグ設定処理のフローチャートである。
【図２３】図２３は、実施例１に係るクエリ実行プランの他の例を説明する図である。
【図２４】図２４は、実施例１に係る結果送信処理のフローチャートである。
【図２５】図２５は、実施例１に係るスレッド実行処理のフローチャートである。
【図２６】図２６は、実施例１に係るタスク実行処理のフローチャートである。
【図２７】図２７は、実施例１に係るコンテキスト検索処理のフローチャートである。
【図２８】図２８は、実施例１に係るクエリ実行プラン実行処理のフローチャートである。
【図２９】図２９は、実施例１に係るＤＢページ取得処理のフローチャートである。
【図３０】図３０は、実施例１に係る新規タスク追加処理のフローチャートである。
【図３１】図３１は、実施例１に係るコンテキスト共有判定処理のフローチャートである。
【図３２】図３２は、実施例１に係るコンテキスト登録処理のフローチャートである。
【図３３】図３３は、実施例１に係るタスク生成処理のフローチャートである。
【図３４】図３４は、変形例に係る負荷分散処理のフローチャートである。
【図３５】図３５は、実施例２に係る計算機システムの構成を示す。
【発明を実施するための形態】
【実施例１】
【００１４】
まず、実施例１の概要について説明する。
【００１５】
図１は、実施例１に係るＤＢＭＳの概要を説明する図である。
【００１６】
ＤＢＭＳ１４１は、クライアント通信制御部１４２と、クエリ実行プラン１４３と、クエリ実行部１４４と、実行タスク管理部１４５と、スレッド管理部１４６と、ＤＢバッファ管理部１４７とを有する。クエリ実行部１４４は、クエリ実行プラン実行部１５１と、コンテキスト管理部１５２と、コンテキスト共有判定部１５３とを有する。
【００１７】
ＤＢＭＳ１４１（クエリ実行部１４４）は、クエリの実行において、処理ステップを実行するためのタスクを動的に生成し、動的に生成されたタスクを実行する。具体的には、例えば、ＤＢＭＳ１４１（クエリ実行部１４４）は、クエリの実行において、（ａ）処理ステップを実行するためのタスクを生成すること、（ｂ）生成されたタスクを実行することで、前記タスクに対応した処理ステップに必要なデータを読み出すためにＤＢへデータ読出し要求を発行すること、（ｃ）上記（ｂ）で実行されたタスクに対応したＮ番目の処理ステップの実行結果に基づき（Ｎ＋１）番目の処理ステップを実行する場合には、前記実行結果に基づくタスクを新たに生成すること（Ｎは１以上の整数）、及び、（ｄ）前記新たに生成したタスクについて上記（ｂ）及び上記（ｃ）を行うこと、を行い、上記（ｂ）及び（ｄ）において、２以上の実行可能なタスクが存在する場合には、それら２以上のタスクのうちの少なくとも２つのタスクを並行して実行することができる。

【００１８】
ＤＢＭＳ１４１（クエリ実行部１４４）は、タスクを実行する際に、オペレーティングシステム（ＯＳ）が提供するスレッド（カーネルスレッド）を複数利用し、それら複数のスレッドが、それぞれ、１又は複数のプロセッサが有する１又は複数のプロセッサコアにより実行される。プロセッサコアがスレッドを実行することで、スレッドに割当たったタスクを実行する。以下、プロセッサコアがタスクを実行する、或いは、ＤＢＭＳ１４１がタスクを実行する、のような表現は、プロセッサコアがスレッドを実行することにより前記スレッドに割当たったタスクを実行することを意味する。
【００１９】
ＤＢＭＳ１４１は、クライアント通信制御部１４２を介して、クエリを受信する。クエリ実行プラン生成部１４３は、受信したクエリを実行するためのクエリ実行プランＰＬを生成する。クエリ実行プラン実行部１５１は、生成されたクエリ実行プランＰＬを実行する。スレッド管理部１４６は、ＤＢＭＳ１４１が構築された計算機におけるプロセッサのプロセッサコアにて実行されるスレッドを複数管理する。実行タスク管理部１４５は、スレッドで実行するタスクを管理する。本実施例では、実行タスク管理部１４５は、スレッド１つに対して、複数のタスクを割り当てることができる。これにより、スレッドの管理に要するオーバヘッドを低減することができる。
【００２０】
コンテキスト管理部１５２は、タスクを実行する際に利用するコンテキストを管理する。コンテキストとしては、複数のスレッドから利用できるように管理するスレッド間共有のコンテキストと、１つのスレッドから利用できるように管理するスレッド間非共有のコンテキストがある。スレッド間非共有のコンテキストを利用できるスレッドは、他のスレッドに比べ前記コンテキストを優先的に利用する。
【００２１】
図１の例では、コンテキスト＃０とコンテキスト＃１がスレッド間共有のコンテキストであり、コンテキスト＃２、コンテキスト＃３、及びコンテキスト＃４がスレッド間非共有のコンテキストである。スレッド＃１およびスレッド＃２およびスレッド＃３に割当たったタスクを実行する際に、コンテキスト＃０とコンテキスト＃１を利用する。スレッド＃１に割当たったタスクを実行する際に、コンテキスト＃２を利用する。スレッド＃２に割当たったタスクを実行する際に、コンテキスト＃３を利用する。コンテキスト＃４は、スレッド＃３に割当たったタスクを実行する際に、コンテキスト＃４を利用する。
【００２２】
コンテキストを、スレッド間共有とするか、スレッド間非共有とするかは、コンテキスト共有判定部１５３による判定結果に基づいて決定する。例えば、コンテキスト共有判定部１５３は、クエリ実行プランの先頭の処理ステップに関わるコンテキストを、スレッド間共有のコンテキストとして良い。また、コンテキスト共有判定部１５３は、クエリ実行プランが並行実行可能な複数の処理ブロックにより構成されている場合には、各処理ブロックの最初の処理ステップに関わるコンテキストを、スレッド間共有のコンテキストとしても良い。また、コンテキスト共有判定部１５３は、１つの処理ブロックにおいて後続の処理ステップ数が所定数以上である処理ステップに関わるコンテキストを、スレッド間共有のコンテキストとして良い。処理ブロックは、１以上の処理ステップで構成されていて良い。並行実行可能な複数の処理ブロックにより構成されているクエリ実行プランの一例は、後述する。
【００２３】
図２は、実施例１に係るＤＢ管理システムにおけるクエリの実行を説明する図である。なお、同図においては、上から下方向に時間の経過を表現している。
【００２４】
まず、スレッド管理部１４６が、スレッド＃１、スレッド＃２、スレッド＃３を生成する。これらスレッド＃１～＃３は、例えば、異なるプロセッサコアにより並行して実行可能である。クエリ実行プラン実行部１５１（具体的には、１つのプロセッサコア）が、ク

エリの実行を開始するコンテキスト＃０を生成し、タスク＃１を生成する。コンテキスト＃０は、クエリ実行プランの先頭である処理ステップ＃１に関するコンテキストであるのでスレッド間共有となる。クエリ実行プラン実行部１５１は、タスク＃１をスレッド＃１に割り当てる。スレッド＃１は、コンテキスト＃０を利用し、タスク＃１を実行する。タスク＃１を実行すると、コンテキスト＃１を生成し、タスク＃２とタスク＃３を生成する。コンテキスト＃１は、クエリ実行プランの先頭である処理ステップ＃１に関するコンテキストであるので、スレッド間共有となる。スレッド＃１は、タスク＃２をスレッド＃２に割り当て、タスク＃３をスレッド＃３に割り当てる。スレッド＃２は、コンテキスト＃１を利用し、タスク＃２を実行する。スレッド＃３は、コンテキスト＃１を利用し、タスク＃３を実行する。

【００２５】

スレッド＃１がタスク＃１を実行し、ＤＢへのアクセス処理を実行する。その結果、新たなコンテキストを生成する。例えば、処理ステップ＃３に関するコンテキスト＃２を生成する。コンテキスト＃２の処理ステップは、クエリ実行プランの先頭ではないため、スレッド間非共有となっている。すなわち、コンテキスト＃２は、基本的には、スレッド＃１によって利用される。スレッド＃１がタスク＃４を生成し、スレッド＃１に割り当てる。スレッド＃１は、コンテキスト＃２を利用し、タスク＃４を実行する。

【００２６】

スレッド＃２がタスク＃２を実行し、ＤＢへのアクセス処理を実行する。その結果、新たなコンテキストを生成する。例えば、処理ステップ＃３に関するコンテキスト＃３を生成する。コンテキスト＃３の処理ステップは、クエリ実行プランの先頭ではないため、スレッド間非共有となっている。すなわち、コンテキスト＃３は、基本的には、スレッド＃２によって利用される。スレッド＃２がタスク＃５とタスク＃６を生成し、スレッド＃２に割り当てる。スレッド＃２は、コンテキスト＃３を利用し、タスク＃５を実行する。さらに、スレッド＃２は、コンテキスト＃３を利用し、タスク＃６を実行する。

【００２７】

スレッド＃３がタスク＃３を実行し、ＤＢへのアクセス処理を実行する。その結果、新たなコンテキストを生成する。例えば、処理ステップ＃３に関するコンテキスト＃４を生成する。コンテキスト＃４の処理ステップは、クエリ実行プランの先頭ではないため、スレッド間非共有となっている。すなわち、コンテキスト＃４は、基本的には、スレッド＃３によって利用される。スレッド＃３がタスク＃７とタスク＃８を生成し、スレッド＃３に割り当てる。スレッド＃３は、コンテキスト＃４を利用し、タスク＃７を実行する。さらに、スレッド＃３は、コンテキスト＃４を利用し、タスク＃８を実行する。

【００２８】

このように、一つのスレッドが複数のタスクを実行し、複数の前記スレッドでクエリを実行する。その結果、複数のプロセッサコアを使い、かつ、スレッドの管理オーバヘッドが小さくなるため、クエリの実行時間が短縮できる。

【００２９】

以下、実施例１を詳細に説明する。

【００３０】

図３は、実施例１に係る計算機システムの構成図である。

【００３１】

計算機システムは、計算機１００と、外部ストレージ装置２００とを有する。計算機１００と、外部ストレージ装置２００とは、通信ネットワーク３００を介して接続されている。通信ネットワーク３００を介した通信のプロトコルとしては、例えば、ＦＣ（Fibre Channel）、ＳＣＳＩ（Small Computer System Interface）、ＩＢ（Infini Band）、又は、ＴＣＰ／ＩＰ（Transmission Control Protocol／Internet Protocol）が採用されて良い。

【００３２】

計算機１００は、例えば、パーソナルコンピュータ、ワークステーション又はメインフ

レームである。計算機１００は、ネットワークアダプタ１１０、プロセッサ（典型的にはマイクロプロセッサ（例えばＣＰＵ（Central Processing Unit））１２０、ローカル記憶デバイス１３０、及びメモリ１４０を有する。プロセッサ１２０は、コンピュータプログラム、例えば、図示しないＯＳ（Operating System）や、ＤＢＭＳ１４１を実行する。１又は複数のプロセッサ１２０は、１又は複数のプロセッサコアを有する。各プロセッサコアがそれぞれ独立して処理を実行することができるようになっている。プロセッサコアは、メモリ１４０よりもアクセスレイテンシの短いキャッシュを持つ。プロセッサコアは、メモリ１４０に記録されたデータをキャッシュに保持し、当該データの処理を行う。一つのプロセッサコアが同じデータを連続して処理する場合は、キャッシュに保持したデータが利用できるため、別々のプロセッサコアが同じデータを連続して処理した場合に比べ処理時間は短い。本実施例では、基本的には、各プロセッサコアは、ある時点で見ると、スレッド（カーネルスレッド）を１つ実行することができる。メモリ１４０は、プロセッサ１２０によって実行されるプログラムと、プログラムが使用するデータとを一時的に記憶する。本実施例では、メモリ１４０は、ＤＢの管理や関連する一連の処理を行うプログラムであるＤＢＭＳ１４１及びデータを記憶する。メモリ１４１は、ＤＢＭＳ１４１にクエリを発行するためのＡＰ（Application Program）１４８を記憶するようにしても良い。ローカル記憶デバイス１３０は、プログラム、及びプログラムが使用するデータを格納する。ネットワークアダプタ１１０は、通信ネットワーク３００と計算機１００とを接続する。また、プロセッサ１２０は、ネットワークアダプタ１１０及びメモリ１４０等に接続された制御デバイスに含まれている要素で良い。制御デバイスは、プロセッサ１２０の他に、専用ハードウェア回路（例えば、データの暗号化及び／又は復号化を行う回路）を含んで良い。

【００３３】

なお、計算機１００は、性能面や冗長性の観点から、ネットワークアダプタ１１０、プロセッサ１２０、ローカル記憶デバイス１３０、及びメモリ１４０のうちの少なくとも１つの要素を複数備えていても良い。また、計算機１００は、図示しない入力デバイス（例えば、キーボード及びポインティングデバイス）と表示デバイス（例えば液晶ディスプレイ）とを有して良い。入力デバイスと表示デバイスは一体になっていても良い。

【００３４】

計算機１００では、ＤＢＭＳ１４１が、ＤＢＭＳ１４１に対して発行されたクエリを実行する。このクエリは、計算機１００で実行されるＡＰ１４８又は、通信ネットワーク３００に接続された図示しない計算機（クライアント）で実行されるＡＰにより発行される。ＤＢＭＳ１４１は、ＡＰ１４８により発行されたクエリを実行し、前記クエリの実行に伴い、外部ストレージ装置２００に格納されたＤＢ２０６に対するＩ／Ｏ要求を、ＯＳを介して外部ストレージ装置２００に送信する。なお、ＯＳは、仮想化プログラムが作成し実行する仮想マシン上で動作するＯＳであっても良い。

【００３５】

外部ストレージ装置２００は、計算機１００が使用するデータを記憶する。外部ストレージ装置２００は、計算機１００からＩ／Ｏ要求を受信し、Ｉ／Ｏ要求に対応した処理を実行し、処理結果を計算機１００に送信する。

【００３６】

外部ストレージ装置２００は、ネットワークアダプタ２０１、記憶デバイス群２０３及びそれらに接続されたコントローラ２０２を有する。

【００３７】

ネットワークアダプタ２０１は、外部ストレージ装置２００を通信ネットワーク３００に接続する。

【００３８】

記憶デバイス群２０３は、１つ以上の記憶デバイスを含む。記憶デバイスは、不揮発性の記憶媒体であって、例えば、磁気ディスク、フラッシュメモリ、その他半導体メモリである。記憶デバイス群２０３は、ＲＡＩＤ（Redundant ARRAY of Independent Disks

）に従い所定のＲＡＩＤレベルでデータを記憶するグループであっても良い。記憶デバイス群２０３の記憶空間に基づく論理的な記憶デバイス（論理ボリューム）が計算機１００に提供されても良い。記憶デバイス群２０３は、ＤＢ２０６を記憶する。ＤＢ２０６は、１つ以上の表２０４や索引２０５を含む。表は１つ以上のレコードの集合であり、レコードは１つ以上のカラムから構成される。索引は、表の中の１つ以上のカラムを対象に作成されるデータ構造であり、当該索引が対象とするカラムを含む選択条件による表へのアクセスを高速化する。例えば、索引は、対象とするカラムの値毎に前記値を含む表の中のレコードを特定する情報（ＲｏｗＩＤ）を保持するデータ構造であり、Ｂ木構造などが用いられる。ＤＢの表の構成例や表同士の関連性の一例は、後述する。

【００３９】

コントローラ２０２は、例えば、メモリ及びプロセッサを含んでおり、計算機１００からのＩ／Ｏ要求に従って、ＤＢ２０６を記憶した記憶デバイス群２０３にデータを入出力する。例えば、コントローラ２０２は、計算機１００からの書込み要求に従う書込み対象のデータを記憶デバイス群２０３に格納したり、計算機１００からの読出し要求に従う読出し対象のデータを記憶デバイス群２０３から読み出し、前記データを計算機１００に送信したりする。

【００４０】

なお、外部ストレージ装置２００は、性能面や冗長性確保の観点から、コントローラ２０２などの要素を複数備えても良い。また、外部ストレージ装置２００を複数備えても良い。

【００４１】

ＤＢＭＳ１４１は、業務データを含んだＤＢ２０６を管理する。ＤＢＭＳ１４１は、クライアント通信制御部１４２、クエリ実行プラン生成部１４３、クエリ実行部１４４、実行タスク管理部１４５、スレッド管理部１４６、及びＤＢバッファ管理部１４７を含む。

【００４２】

クライアント通信制御部１４２は、通信ネットワーク３００に接続されたクライアントまたはＡＰ１４８との間の通信を制御する。具体的には、クライアント通信制御部１４２は、クライアントまたはＡＰ１４８から発行されたクエリを受信し（受け付け）、クエリの処理結果をクライアントまたはＡＰ１４８に送信する処理を実行する。クエリは、例えばＳＱＬ（Structured Query Language）で記述されている。

【００４３】

クエリ実行プラン生成部１４３は、クライアント通信制御部１４２が受け付けたクエリを実行するために必要な１つ以上の処理ステップを有するクエリ実行プランを生成する。クエリ実行プランは、例えば、クエリの実行の際に行うべき処理ステップの実行順序を木構造で定義した情報であり、メモリ１４０に格納される。クエリ実行プランの一例は、後述する。

【００４４】

ＤＢバッファ管理部１４７は、ＤＢ２０６内のデータを一時的に格納するための記憶領域（ＤＢバッファ）を管理する。ＤＢバッファは、メモリ１４０上に構築される。また、ＤＢバッファは、ローカル記憶デバイス１３０上に構築されても良い。

【００４５】

クエリ実行部１４４は、クエリ実行プラン生成部１４３が生成したクエリ実行プランに従ってクエリを実行し、生成した結果をクエリの発行元に返す。クエリ実行部１４４は、クエリ実行プラン実行部１５１と、コンテキスト管理部１５２と、コンテキスト共有判定部１５３とを有する。

【００４６】

クエリ実行プラン実行部１５１は、クエリ実行プラン内の処理ステップを実行するためのタスクを動的に生成し、スレッドにタスクを割り当て、スレッドがタスクを実行することでクエリを実行する。

【００４７】

コンテキスト管理部１５２は、生成するタスクの実行に必要な情報を含むコンテキストを管理する。ここで、コンテキストは、タスクにおいて実行を開始する処理ステップが、クエリ実行プランが表す１以上の処理ステップのうちのいずれであるかを示す第１の情報と、第１の情報が示す処理ステップに要するデータのアクセス先に関する第２の情報と、タスクにより結果を生成するために必要なデータに関する第３の情報とを含む情報である。コンテキストを管理するための情報であるコンテキスト管理情報の構造については後述する。

【００４８】

コンテキスト共有判定部１５３は、コンテキストを複数のスレッド間で共有するか否かを判定する。

【００４９】

実行タスク管理部１４５は、スレッドが実行するタスクを管理する。タスクは、例えば、ＤＢＭＳ４１２で実装される疑似プロセス又は疑似スレッド（ユーザレベルスレッド）である。なお、タスクは、各処理を関数としてまとめた関数へのポインタ（関数ポインタ）の集合であってもよい。タスクを管理するための情報であるタスク管理情報の構造については、後述する。

【００５０】

スレッド管理部１４６は、クエリを実行するためのスレッドを管理する。ここで、スレッドとは、ＯＳが提供するスレッド（カーネルスレッド）である。前述したように、プロセッサコアが割当たったスレッドを実行することにより、スレッドに割当たったタスクが実行される。なお、スレッドの代わりにプロセスを使用してもよい。

【００５１】

クライアント通信制御部１４２、クエリ実行プラン生成部１４３、クエリ実行部１４４、及びＤＢバッファ管理部１４７の少なくとも１つの処理部が行う処理の少なくとも一部が、ハードウェアで行われても良い。また、本実施例の説明において、処理部が主語になる場合は、実際には前記処理部を実行するプロセッサ１２０によって処理が行われるが、処理部の少なくとも一部がハードウェアで実現されている場合は、プロセッサ１２０に代えて又は加えて、前記ハードウェアも、主語とされ得る。ＤＢＭＳ１４１を実現するコンピュータプログラムは、プログラムソースから計算機１００にインストールされて良い。プログラムソースは、例えば、計算機１００が読み取り可能な記憶メディアで良いし、他の計算機でも良い。

【００５２】

また、図３に示したＤＢＭＳ１４１の構成は、一例である。例えば、或る処理部が複数の処理部に分割されたり、複数の処理部の機能を統合した１つの処理部が構築されたりしても良い。

【００５３】

図４は、実施例１に係るＤＢの表及び索引の定義を説明する図である。

【００５４】

ＤＢ２０６は、表２０５として、例えば、カラムｃ１及びカラムｃ２を含むＰａｒｔ表と、カラムｃ３及びカラムｃ４を含むＬｉｎｅｉｔｅｍ表とを有する。また、ＤＢ２０６は、索引２０４として、カラムｃ１の値に基づいたＰａｒｔ表に関する索引（Ｐａｒｔ索引）と、カラムｃ３の値に基づいたＬｉｎｅｉｔｅｍ表に関する索引（Ｌｉｎｅｉｔｅｍ索引）とを有する。

【００５５】

図５は、実施例１に係るＤＢのＰａｒｔ表の一例を示す図である。

【００５６】

ＤＢ２０６のＰａｒｔ表は、論理的には、例えば、カラムｃ１の値と、対応するカラムｃ２の値とを対応付けた表となっている。

【００５７】

図６は、実施例１に係るＤＢのＬｉｎｅｉｔｅｍ表の一例を示す図である。

【００５８】
ＤＢ２０６のＬｉｎｅｉｔｅｍ表は、例えば、カラムｃ３の値と、対応するカラムｃ４の値とを対応付けた表となっている。
【００５９】
図７は、実施例１に係るＤＢにおけるｐａｒｔ索引及びｐａｒｔ表のデータ構造の一例を説明する図である。
【００６０】
Ｐａｒｔ索引は、例えば、カラムｃ１の値に基づいて、対応するカラムｃ２の値を格納するｐａｒｔ表のページ及びスロットを検索するためのＢ木構造となっている。ページとは、ＤＢ２０６に対する入出力における最小のデータ単位である。本実施例では、Ｐａｒｔ索引は、ページＰを階層構造として管理している。Ｐａｒｔ索引においては、最下位のページであるリーフページと、リーフページの上位のページである上位ページとがある。ここで、上位ページの中の最上位のページをルートページということとする。
【００６１】
Ｐａｒｔ索引のルートページ（ページＰ１）には、一つ下の階層のページに対するポインタと、当該一つ下の階層のページが管理対象とするカラムｃ１の値の最大値とを対応付けたエントリが１以上設けられる。例えば、ページＰ１には、「１００」以下のカラムｃ１の値に対する対応関係を管理するページＰ２へのポインタと、「１００」より大きく「２００」以下のカラムｃ１の値に対する対応関係を管理するページＰ３へのポインタと、「２００」より大きく「３００」以下のカラムｃ１の値に対する対応関係を管理するページＰ４へのポインタとが格納される。同様に、上位ページにおいては、それぞれのページの一つ下の階層のページに対するポインタと、当該１つ下の階層のページに管理されているカラムｃ１の値の最大値とを対応付けたエントリが１以上設けられる。
【００６２】
一方、リーフページには、カラムｃ１の値と、当該値に対応するカラムｃ２の値を格納するＰａｒｔ表における格納位置（例えば、Ｐａｒｔ表のページ番号及びの当該ページ中のスロット番号）とを対応付けたロー（行：レコード）を１以上格納する。
【００６３】
例えば、リーフページであるページＰ８には、カラムｃ１の値「１１０」に対応するカラムｃ２の値が格納されているページ及びスロットの番号を含むローと、カラムｃ１の値「１３０」に対応するカラムｃ２の値が格納されているページ及びスロットの番号を含むローとが格納される。例えば、カラムｃ１の値「１３０」に対応するカラムｃ２の値が格納されているページ及びスロットの番号を含むローとしては、ページＰ１００のスロット２、ページＰ１２０のスロット１、ページＰ２００のスロット４とを示すローが格納される。従って、カラムｃ１の値「１３０」に対応するカラムｃ２の値は、Ｐａｒｔ表のページＰ１００のスロット２のレコードから「ｉｄ１３１」となり、また、Ｐａｒｔ表のページ１２０のスロット１のレコードから「ｉｄ１３２」となり、Ｐａｒｔ表のページ２００のスロット４のレコードから「ｉｄ１３３」となる。
【００６４】
図８は、実施例１に係るＤＢのクエリの一例を示す図である。
【００６５】
図８に示すクエリは、図４～図７に示す構造のＤＢ２０６に対するクエリの一例である。図８に示すクエリは、Ｐａｒｔ表及びＬｉｎｅｉｔｅｍ表から、カラムｃ１の値が「１３０」であり、且つカラムｃ２の値とカラムｃ３の値とが同じであるものについて、カラムｃ１の値とカラムｃ４の値とを抽出することを意味している。
【００６６】
図９は、実施例１に係るクエリ実行プランの一例を示す図である。
【００６７】
同図に示すクエリ実行プランは、ＤＢＭＳ１４１が図８に示すクエリを受け付けた場合に、クエリ実行プラン生成部１４３により生成されるクエリ実行プランを示している。

【００６８】
図８に示すクエリに対応するクエリ実行プランは、図９に示すように、Ｐａｒｔ索引による索引検索を行う処理ステップ＃１と、Ｐａｒｔ表からレコードを取得する処理ステップ＃２と、Ｌｉｎｅｉｔｅｍ索引による索引検索を行う処理ステップ＃３と、Ｌｉｎｅｉｔｅｍ表からレコードを取得する処理ステップ＃４と、これらの結果をネストループ結合する処理ステップ＃５とを含む。
【００６９】
図１０は、実施例１に係るタスク管理情報のデータ構造の一例を示す図である。
【００７０】
タスク管理情報は、メインデータ構造体７１を有する。メインデータ構造体７１は、複数のスレッドを特定するスレッド特定情報（例えば、スレッド番号）と、スレッドが実行するタスクを管理するリスト管理構造体７２へのポインタを、スレッドごとに対応付けて記憶する。
【００７１】
リスト管理構造体７２は、対応するスレッドにおける実行可能なタスクを管理するための実行可能リスト７２ａと、対応するスレッドにおける実行待ち状態であるタスクを管理するための待ちリスト７２ｂとを記憶する。実行可能リスト７２ａは、対応するスレッドにおいて実行可能なタスクに関する実行状態情報（タスク実行状態情報）７３へのポインタを有する。また、タスク実行状態情報７３は、対応するスレッドにおける実行可能な他のタスクに関するタスク実行状態情報７３へのポインタを有する。
【００７２】
図１０においては、例えば、スレッド＃２における実行可能なタスクに関するタスク実行状態情報７３としては、タスク＃２に対する実行状態情報（タスク＃２実行状態情報）が管理され、スレッド＃２における実行待ち状態であるタスクに関するタスク実行状態情報７３としては、タスク＃５及びタスク＃６のタスク実行状態情報７３が管理されている。なお、複数のスレッド間においてタスクの偏りがある場合には、タスク（すなわち、タスク実行状態情報７３）を別のスレッドのリストに移動させるようにしてもよい。
【００７３】
なお、本実施例では、スレッド毎に、実行可能リスト及び待ちリストを管理するようにしているが、実行可能リスト及び待ちリストを複数のスレッド間で共有しても良い。また、処理ステップごとに、実行可能リスト及び待ちリストを管理するようにしても良い。
【００７４】
図１１は、実施例１に係るタスク実行状態情報の一例を示す図である。
【００７５】
タスク実行状態情報７３は、ワーク領域７３ａと、処理ステップ７３ｂと、処理ステップ実行状態７３ｃとを格納する。ワーク領域７３ａには、ワーク領域を示すポインタが格納される。処理ステップ７３ｂには、対応するタスクにより実行する処理ステップを識別する情報、例えば、処理ステップ番号が格納される。処理ステップ実行状態７３ｃには、対応する処理ステップの実行状態情報（処理ステップ実行状態情報）７４が格納される。処理ステップ実行状態情報７４の具体例については、後述する。
【００７６】
図１２は、実施例１に係る処理ステップ実行状態情報の第１の例を示す図である。図１２は、索引検索における上位ページを使用するタスクについての処理ステップ実行状態情報を示す。
【００７７】
処理ステップ実行状態情報７４Ａは、検索条件７４ａと、ページ番号７４ｂと、スロット番号７４ｃとを含む。検索条件７４ａには、検索条件を格納する。同図の例では、検索条件７４ａには、クエリに含まれる検索条件であるｋｅｙ値の範囲「１１５以上ｋｅｙ以上１９５」を格納する。ページ番号７４ｂには、タスクの処理で使用する上位ページの番号（ページ番号）を格納する。スロット番号７４ｃには、タスクの処理で使用するページ

におけるスロットの番号（スロット番号）を格納する。

【００７８】

図１３は、実施例１に係る処理ステップ実行状態情報の第２の例を示す図である。図１３は、索引検索におけるリーフページを使用するタスクについての処理ステップ実行状態情報を示す。

【００７９】

処理ステップ実行状態情報７４Ｂは、検索条件７４ｄと、ページ番号７４ｅと、スロット番号７４ｆと、処理ローＩＤ数７４ｇとを含む。検索条件７４ｄには、検索条件を格納する。同図の例では、検索条件７４ｄには、検索条件であるｋｅｙ値の範囲「１１５以上Ｋｅｙ以上１９５」を格納する。ページ番号７４ｅには、タスクの処理で使用するリーフページのページ番号を格納する。スロット番号７４ｆには、タスクの処理で使用するページにおけるスロットのスロット番号を格納する。処理ローＩＤ番号７４ｇには、対応するタスクで処理するスロット内のローのＩＤ番号（処理ローＩＤ番号）を格納する。

【００８０】

図１４は、実施例１に係る処理ステップ実行状態情報の第３の例を示す図である。図１４は、レコード取得を行うタスクについての処理ステップ実行状態情報を示す。

【００８１】

処理ステップ実行状態情報７４Ｃは、ページ番号７４ｈと、スロット番号７４ｉとを含む。ページ番号７４ｈには、タスクの処理で使用するページのページ番号を格納する。スロット番号７４ｉには、タスクの処理で使用するページにおけるスロットのスロット番号を格納する。

【００８２】

図１５は、実施例１に係るコンテキスト管理情報のデータ構造の一例を説明する図である。

【００８３】

コンテキスト管理情報８０は、管理リストのメイン構造体８１と、複数のコンテキスト８２を含む。メイン構造体８１には、コンテキスト８２へのポインタを格納する。また、各コンテキスト８２には、他のコンテキスト８２へのポインタを格納する。本実施例では、タスクの処理でコンテキスト８２を利用する場合には、タスクを実行しているスレッドが各コンテキスト８２を単位として排他（ロック）をかける。ロック状態のコンテキストは他のスレッドからは利用できない。

【００８４】

また、コンテキスト管理情報８０は、クエリを実行するスレッドに対応する検索テーブル（スレッド向け検索テーブル）８３、８４、８５等を記憶する。スレッド＃１向け検索テーブル８３は、スレッド＃１が利用可能なコンテキスト８２へのポインタを管理する。スレッド＃２向け検索テーブル８４は、スレッド＃２が利用可能なコンテキスト８２へのポインタを管理する。スレッド＃３向け検索テーブル８５は、スレッド＃３が利用可能なコンテキスト８２へのポインタを管理する。

【００８５】

図１６は、実施例１に係るスレッド＃１向け検索テーブルの一例を示す図である。図１７は、実施例１に係るスレッド＃２向け検索テーブルの一例を示す図である。図１８は、実施例１に係るスレッド＃３向け検索テーブルの一例を示す図である。

【００８６】

スレッド＃１向け検索テーブル８３は、スレッド＃１が利用可能なコンテキストへのポインタを管理するテーブルであり、各処理ステップに関わるコンテキストへのポインタをリストで管理する。図１６に示すように、処理ステップ＃１に関わるコンテキストへのポインタ８３ａと、処理ステップ＃２に関わるコンテキストへのポインタ８３ｂと、処理ステップ＃３に関わるコンテキストへのポインタ８３ｃと、処理ステップ＃４に関わるコンテキストへのポインタ８３ｄとを含む。本実施例では、ポインタ８３ａにはコンテキスト＃１へのポインタを格納し、ポインタ８３ｃにコンテキスト＃２へのポインタを格納する

。

【００８７】

スレッド＃２向け検索テーブル８４は、スレッド＃２が利用可能なコンテキストへのポインタを管理するテーブルであり、各処理ステップに関わるコンテキストへのポインタをリストで管理する。図１７に示すように、処理ステップ＃１に関わるコンテキストへのポインタ８４ａと、処理ステップ＃２に関わるコンテキストへのポインタ８４ｂと、処理ステップ＃３に関わるコンテキストへのポインタ８４ｃと、処理ステップ＃４に関わるコンテキストへのポインタ８４ｄとを含む。本実施例では、ポインタ８４ａにコンテキスト＃１へのポインタを格納し、ポインタ８４ｃにコンテキスト＃３へのポインタを格納する。

【００８８】

スレッド＃３向け検索テーブル８５は、スレッド＃３が利用可能なコンテキストへのポインタを管理するテーブルであり、各処理ステップに関わるコンテキストへのポインタをリストで管理する。図１８に示すように、処理ステップ＃１に関わるコンテキストへのポインタ８５ａと、処理ステップ＃２に関わるコンテキストへのポインタ８５ｂと、処理ステップ＃３に関わるコンテキストへのポインタ８５ｃと、処理ステップ＃４に関わるコンテキストへのポインタ８５ｄとを含む。本実施例では、ポインタ８５ａにコンテキスト＃１へのポインタを格納し、ポインタ８５ｃにコンテキスト＃４へのポインタを格納する。

【００８９】

スレッド＃１向け検索テーブル８３、スレッド＃２向け検索テーブル８４、及びスレッド＃３向け検索テーブル８５によると、コンテキスト＃１は、スレッド＃１、スレッド＃２、又はスレッド＃３において利用可能である。一方、コンテキスト＃２は、スレッド＃１において利用可能である。コンテキスト＃３は、スレッド＃２において利用可能であり、コンテキスト＃４は、スレッド＃３において利用可能である。ここで、コンテキスト８２へのポインタが複数のスレッド向け検索テーブルに登録してある状態を、コンテキストがスレッド間共有であるといい、コンテキストへのポインタが特定の一つのスレッド向け検索テーブルに登録してある状態をコンテキストがスレッド間非共有であるという。ここで、あるスレッドを実行するプロセッサコアが当該スレッド向けのスレッド向け検索テーブルを用いて利用可能なコンテキストのことを、当該スレッドが利用可能なコンテキストという。

【００９０】

この状態では、コンテキスト＃１はスレッド＃１、スレッド＃２、スレッド＃３が利用し、コンテキスト＃２はスレッド＃１が利用し、コンテキスト＃３はスレッド＃２が利用し、コンテキスト＃４はスレッド＃３が利用する。この状態の下では、スレッド間非共有のコンテキストは一つのスレッドが連続利用するため、プロセッサコアのキャッシュによりコンテキスト利用に伴う処理の処理時間が短くできる。

【００９１】

なお、本実施例では、利用可能なコンテキストがスレッド間で偏った場合に、スレッドが実行するタスクの量を均等にする目的で、他のスレッド向けのスレッド向け検索テーブルを参照してもよい。具体的には、スレッドが利用可能なコンテキストがない場合、他のスレッド向けのスレッド向け検索テーブルを参照することにより、他のスレッドがスレッド間非共有のコンテキスト（コンテキスト＃２、コンテキスト＃３、コンテキスト＃４）を利用する。例えば、スレッド＃１においてスレッド＃１が利用可能なコンテキストがなくなった場合、スレッド＃１がスレッド＃２向け検索テーブルやスレッド＃３向け検索テーブルを参照し、コンテキスト＃３やコンテキスト＃４をスレッド＃１に割当たったタスクが利用する。

【００９２】

図１９は、実施例１に係るコンテキストの一例を示す図である。

【００９３】

コンテキスト８２は、開始ステップ８２ａと、中間結果８２ｂと、実行状態８２ｃと、生成可能数８２ｄとを含む。開始ステップ８２ａには、対応する処理ステップの番号を格

納する。中間結果８２ｂには、対応する処理ステップを実行するタスクに必要な中間結果を格納するワーク領域を示すポインタを格納する。ここで、中間結果とは、クエリの結果を生成するために必要な取得済みのデータである。実行状態８２ｃには、対応する処理ステップにおけるタスクの実行状態、例えば、次に実行するタスクの処理の内容を特定する情報（例えば、ページ番号８２０、スロット番号８２１、及び処理ローＩＤ番号８２２）を格納する。ここで、ページ番号８２０には、次のタスクの処理で使用するリーフページのページ番号を格納する。スロット番号８２１には、次のタスクの処理で使用するページにおけるスロット番号を格納する。処理ローＩＤ番号８２２には、次のタスクの処理で使用するスロット内のローのＩＤ番号（処理ローＩＤ番号）を格納する。生成可能数８２ｄには、対応する処理ステップにおいて、さらに生成することのできるタスクの数（タスク生成可能数）を格納する。このタスク生成可能数は、論理的に分岐する処理の数の内で、タスクとして生成されていない処理の数である。例えば、図７に示すＰａｒｔ索引による索引検索でキー値「１３０」を条件としている場合に、ページＰ８においては、キー値「１３０」に対応するエントリとしては、ローＩＤが３個あるので、全体として３つのローＩＤを用いてＰａｒｔ表のレコードを取得する３つのタスクを生成することができる。ここで、一つのローＩＤはコンテキストを生成するタスクで処理するので、生成したコンテキストから残りの２つのタスクを生成することができる。このため、タスク生成可能数は、「２」となる。

【００９４】

図２０は、実施例１に係るクエリ受付時処理のフローチャートである。

【００９５】

クエリ受付時処理においては、クライアント通信制御部１４２が、ＡＰ１４８からクエリを受け付けると（ステップＳ１）、受け付けたクエリをクエリ実行プラン生成部１４３に渡し、クエリ実行プラン生成部１４３がクエリ実行プラン生成処理（図２１参照）を実行する（ステップＳ２）。

【００９６】

クエリ実行プラン生成処理の実行後に、スレッド管理部１４６がスレッドを生成する（ステップＳ３）。ここで、生成するスレッド数は、任意の数であってよく、例えば、プロセッサ１２０のプロセッサコアの数と同じ数であっても良い。ここで、スレッドが動作するプロセッサコアをスレッドごとに特定のプロセッサコアに指定しても良い。すなわち、プロセッサアフィニティを設定するようにしても良い。例えば、プロセッサコアの数と同数のスレッドを生成し、各プロセッサコアでいずれか一つのスレッドが実行されるように設定しても良い。このようにすると、各スレッドによる処理の効率が良い。ここで、スレッドを生成する方法としては、ＯＳが提供するスレッド生成のインタフェース（関数）、具体的には、pthread_create()を利用する方法がある。

【００９７】

次いで、クエリ実行プラン実行部１５１がクエリの実行を開始するコンテキストを生成し、前記コンテキストを利用して処理を行うタスクを生成し、いずれか一つのスレッドに割り当てる（ステップＳ４）。例えば、スレッド管理部１４６が最初に作成したスレッドにタスクを割り当てる。これにより、以降において、プロセッサ１２０のプロセッサコアがスレッドを実行し、前記スレッドに割り当てられたタスクを前記スレッドが実行する。

【００９８】

図２１は、実施例１に係るクエリ実行プラン生成処理のフローチャートである。

【００９９】

クエリ実行プラン生成処理は、図２０に示すクエリ受付時処理のステップＳ２に対応する処理である。クエリ実行プラン生成部１４３は、クライアント通信制御部１４２から渡されたクエリからクエリ実行プランを生成する（ステップＳ５）。例えば、図８に示すクエリを受け付けた場合には、図９に示すクエリ実行プランを生成する。次いで、クエリ実行プラン生成部１４３は、スレッド間共有フラグ設定処理（図２２参照）を実行し（ステップＳ６）、クエリ実行プラン生成処理を終了する。

【０１００】
図２２は、実施例１に係るスレッド間共有フラグ設定処理のフローチャートである。
【０１０１】
スレッド間共有フラグ設定処理は、図２１に示すクエリ実行プラン生成処理のステップＳ６に対応する処理である。スレッド間共有フラグ設定処理は、クエリ実行プランにおける所定の処理ステップに対して、当該処理ステップに関わるコンテキストをスレッド間共有とすべきことを示すスレッド間共有フラグを設定するための処理である。
【０１０２】
クエリ実行プラン生成部１４３は、木構造となっているクエリ実行プランを辿るために、ポインタを移動させながら処理を行う。クエリ実行プランの最初の処理ステップにポインタを設定する（ステップＳ１１）。次いで、クエリ実行プラン生成部１４３は、クエリ実行プランにポインタが指し示す処理ステップがあるか否かを判定する（ステップＳ１２）。この結果、ポインタが指し示す処理ステップがない場合（ステップＳ１２で「ない」）には、クエリ実行プランの全ての処理ステップを対象に処理をしたことを意味するので、クエリ実行プラン生成部１４３は、スレッド間共有フラグ設定処理を終了する。一方、クエリ実行プランの処理ステップにポインタが指し示す処理ステップがある場合（ステップＳ１２で「ある」）には、クエリ実行プラン生成部１４３は、処理ステップが、処理ブロックの先頭であるか否かを判定する（ステップＳ１３）。
【０１０３】
ここで、処理ブロックとは、クエリ実行プランにおける逐次的に実行しなければならない１つ以上の処理ステップを、並行して実行できる集合に区分した場合における当該集合のことをいう。例えば、図９に示すクエリ実行プランには、１つの処理ブロックが含まれる。ここで、処理ブロックについて、他のクエリ実行プランを用いて説明する。
【０１０４】
図２３は、実施例１に係るクエリ実行プランの他の例を説明する図である。
【０１０５】
図２３に示すクエリ実行プランは、Ｐａｒｔ索引による索引検索を行う処理ステップ＃１と、Ｐａｒｔ表からレコードを取得する処理ステップ＃２と、Ｌｉｎｅｉｔｅｍ表に対してテーブルスキャンを実行する処理ステップ＃３と、処理ステップ＃２及び処理ステップ＃３の結果をハッシュ結合する処理ステップ＃４とを有する。ここで、処理ステップ＃１及び処理ステップ＃２と、処理ステップ＃３とは並行して実行可能な処理である。このクエリ実行プランは、処理ステップ＃１及び処理ステップ＃２が含まれる処理ブロック＃１と、処理ステップ＃３及び処理ステップ＃４が含まれる処理ブロック＃２とを含む。このクエリ実行プランにおいては、処理ステップ＃１と、処理ステップ＃３とが処理ブロックの先頭の処理ステップである。
【０１０６】
なお、このようなクエリ実行プランの他に、例えば、副問合せや、導出表が含まれるクエリに対応するクエリ実行プランにおいても、複数の処理ブロックが含まれる。
【０１０７】
図２２の説明に戻り、ステップＳ１３の結果、処理ステップが処理ブロックの先頭である場合（ステップＳ１３で「はい」）には、クエリ実行プラン生成部１４３は、当該処理ステップに対して、スレッド間共有フラグを設定する（ステップＳ１４）。例えば、図９に示すクエリ実行プランでは、処理ステップ＃１にスレッド間共有フラグを設定する。図２３に示すクエリ実行プランでは、処理ステップ＃１と処理ステップ＃３にスレッド間共有フラグを設定する。そして、処理をステップＳ１５に進める。ここで、処理ステップが処理ブロックの先頭である場合においてコンテキストを複数のスレッドで共有すべきとするのは、処理ブロックの早い段階において、起点となるタスクを複数のスレッドに分散させるためである。
【０１０８】
一方、ステップＳ１３の結果、処理ステップが処理ブロックの先頭でない場合（ステッ

プＳ１３で「いいえ」）には、クエリ実行プラン生成部１４３は、処理をステップＳ１５に進める。

【０１０９】

ステップＳ１５では、クエリ実行プラン生成部１４３が次の処理ステップにポインタを移動させ、処理をステップＳ１２に進む。

【０１１０】

なお、図２２に示すスレッド間共有フラグ設定処理においては、処理ブロックの先頭の処理ステップについて、スレッド間共有フラグを設定するようにしていたが、例えば、処理ブロックにおける後続の処理ステップの数が所定数以上ある処理ステップについて、スレッド間共有フラグを設定するようにしても良い。

【０１１１】

図２４は、実施例１に係る結果送信処理のフローチャートである。

【０１１２】

結果送信処理は、クライアント通信制御部１４２がクエリを受け付けた後に、クライアント通信制御部１４２により開始される。クライアント通信制御部１４２は、クエリ実行部１４４における受け付けたクエリの結果の有無を確認する（ステップＳ２１）。

【０１１３】

この結果、クエリの結果がある場合（ステップＳ２１で「有」）には、クライアント通信制御部１４２は、クエリ実行部１４４からクエリの結果を取得し（ステップＳ２２）、クエリの発行元であるＡＰ１４８に対してクエリの結果を送信する（ステップＳ２６）。

【０１１４】

一方、クエリの結果がない場合（ステップＳ２１で「無」）には、クライアント通信制御部１４２は、クエリ実行部１４４のクエリ終了フラグがクエリの終了を示す「終了」であるか、クエリが終了していないことを示す「未終了」であるかを判定する（ステップＳ２３）。この結果、クエリ終了フラグが「終了」である場合（ステップＳ２３で「終了」）には、結果にＮＯＲＯＷ（該当レコードなし）を設定し（ステップＳ２４）、クエリの発行元であるＡＰ１４８に対してクエリの結果を送信する（ステップＳ２６）。

【０１１５】

一方、クエリ実行部１４４のクエリ終了フラグがクエリの未終了を示す「未終了」である場合（ステップＳ２３で「未終了」）には、クライアント通信制御部１４２は、クエリ実行部１４４が結果を生成するのを所定時間待って（ステップＳ２５）、処理をステップＳ２１に進める。

【０１１６】

図２５は、実施例１に係るスレッド実行処理のフローチャートである。

【０１１７】

スレッド実行処理は、プロセッサ１２０のプロセッサコアが、図２０のステップＳ３において生成されたスレッドを実行することにより実現される。なお、複数のスレッドが存在する場合には、別のプロセッサコアが別のスレッドに対するスレッド実行処理を並行して行うことができる。

【０１１８】

プロセッサコアは、対応するスレッドにおいて実行するタスクを選択する（ステップＳ３１）。具体的には、プロセッサコアは、実行タスク管理部１４５が管理するタスク管理情報の対応するスレッドの実行可能リストに含まれるタスクを選択する。

【０１１９】

次いで、プロセッサコアは、実行するタスクの有無を判定し（ステップＳ３２）、実行するタスクがない場合（ステップＳ３２で「無」）には、処理をステップＳ３４に進める一方、実行するタスクがある場合（ステップＳ３２で「有」）には、タスクの開始又はタスクの再開を行う（ステップＳ３３）。具体的には次の処理を行う。プロセッサコアは、実行可能リストに含まれるタスクの一つを選ぶ。プロセッサコアは、選んだタスクのタスク実行状態情報を確認し、タスクの開始またはタスクの再開を行う。タスク実行状態情報

の処理ステップが設定していない場合はタスクの処理を開始する。具体的には、タスク実行処理（図２６参照）を実行する。タスク実行状態情報の処理ステップが設定してある場合はタスクの処理を再開する。具体的には、タスクの中断した処理から実行を再開する。本実施例では、図２９のステップＳ６６からの処理から再開する。なお、タスクの実行が終了した後、又はタスクの実行が待ち状態となった後、プロセッサコアは、処理をステップＳ３１に進める。

【０１２０】

ステップＳ３４では、プロセッサコアは、他のスレッドの有無を確認し（ステップＳ３４）、他のスレッドがない場合（ステップＳ３４で「無」）には、クエリ実行部１４４のクエリ終了フラグに終了を設定し（ステップＳ３５）、スレッド実行処理を終了する。これにより、当該スレッドが消滅することとなる。一方、他のスレッドがある場合（ステップＳ３４で「有」）には、プロセッサコアは、スレッド実行処理を終了する。これにより、当該スレッドが消滅することとなる。

【０１２１】

図２６は、実施例１に係るタスク実行処理のフローチャートである。

【０１２２】

タスク実行処理は、図２５のステップＳ３３において、タスクの処理を開始する場合における処理に対応する。このタスク実行処理は、プロセッサコアがスレッドにおけるタスクを実行することにより実現される。

【０１２３】

プロセッサコアは、コンテキスト検索処理（図２７参照）を実行し（ステップＳ３６）、コンテキスト検索処理により検索されたコンテキストの有無を確認する（ステップＳ３７）。この結果、コンテキストがない場合（ステップＳ３７で「なし」）には、実行すべきＤＢへのオペレーションがないことを意味するので、タスク実行処理を終了する。

【０１２４】

一方、コンテキストがある場合（ステップＳ３７で「あり」）には、プロセッサコアは、当該タスクのタスク実行状態情報７３（図１１参照）を設定する（ステップＳ３８）。具体的には、プロセッサコアは、検索されたコンテキスト８２の開始ステップ８２ａの値を、タスク実行状態情報７３の処理ステップ７３ｂにコピーする。コンテキストの中間結果８２ｂのポインタが示すワーク領域のデータを、タスク実行状態情報７３のワーク領域７３ａのポインタが示すワーク領域にコピーする。さらに、プロセッサコアは、コンテキスト８２の実行状態８２ｃの値を、タスク実行状態情報７３の処理ステップ実行状態７３ｃにコピーする。例えば、図１９に示すコンテキスト８２である場合には、タスク実行状態情報７３の処理ステップ実行状態７３ｃには、ページ番号として、コンテキスト８２のページ番号８２０の値「８」が格納され、スロット番号として、コンテキスト８２のスロット番号８２１の値「２」が格納され、処理ローＩＤ番号として、コンテキスト８２の処理ローＩＤ番号８２２の値「２」が格納される。この後、プロセッサコアは、コンテキスト８２の実行状態８２ｃを次のタスクにおける処理内容に対応するように更新する。例えば、図１９に示すコンテキスト８２においては、処理ローＩＤ番号８２２の値を１つ進めて「３」とする。

【０１２５】

このように設定された図１９に示すタスク実行状態情報７３によると、当該タスクの処理で、ページ番号「８」のページ（図７のページＰ８）のスロット番号「２」の処理ローＩＤ番号「２」のローＩＤを参照していることを意味している。この結果、このタスクは、以降の処理において、該当するローＩＤが示すページ「Ｐ１２０」のスロット番号「１」に格納されているレコード（ｉｄ１３２を持つレコード）を参照することから処理を開始することとなる。

【０１２６】

ステップＳ３８の後に、プロセッサコアは、クエリ実行プラン実行処理（図２８参照）を実行する。タスクの処理が終了した場合にステップ３９が終了し、処理をステップＳ３

６に進める。
【０１２７】
図２７は、実施例１に係るコンテキスト検索処理のフローチャートである。
【０１２８】
コンテキスト検索処理は、図２６のステップＳ３６に対応する処理である。プロセッサコアは、実行しているスレッド（自スレッド）向けのスレッド向け検索テーブル（８３、８４、又は８５）のポインタを用いて、コンテキストを探す（ステップＳ４１）。本実施例では、プロセッサコアは、最後の処理ステップから最初の処理ステップの順にコンテキストを探す。次いで、プロセッサコアは、探索できたコンテキストの有無を確認する（ステップＳ４２）。この結果、コンテキストが見つかった場合（ステップＳ４２で「有」）には、コンテキスト検索処理を終了する。一方、コンテキストがない場合（ステップＳ４２で「無」）には、利用可能なコンテキストの数がスレッド間で偏っている可能性があることを意味しているので、自スレッド以外のスレッド（他スレッド）向けのスレッド向け検索テーブルのポインタを用いて、コンテキストを探す（ステップＳ４３）。本実施例では、他スレッド向けのスレッド向け検索テーブルからコンテキストを取得する場合には、プロセッサコアは、最初の処理ステップから最後の処理ステップの順にコンテキストを探す。ここで、最初の処理ステップからコンテキストを探すのは、最初の処理ステップに近いものほど、生成するタスク数が多くなる可能性が高いので、早い段階で各スレッドに負荷を分散することができるためである。ステップＳ４３の後、プロセッサコアは、コンテキスト検索処理を終了する。
【０１２９】
例えば、スレッド＃１でコンテキスト検索処理を実行している場合は、スレッド＃１はスレッド＃１向け検索テーブルでコンテキストを探す。スレッド＃１は、スレッド＃１向け検索テーブルでは処理ステップ＃４から処理ステップ＃１の順に探す。図１６の状態では、スレッド＃１はコンテキスト＃２へのポインタを見つけ、コンテキスト＃２を利用する。コンテキスト＃２がなくなった場合、コンテキスト＃１へのポインタを見つけ、コンテキスト＃１を利用する。さらに、コンテキスト＃１がなくなった場合、スレッド＃１向け検索テーブルには利用可能なコンテキストがない状態となり、スレッド＃１は自スレッド以外のスレッド向けの検索テーブルでコンテキストを探す。この場合、処理ステップ＃１から処理ステップ＃４の順に探す。スレッド＃１は、スレッド＃２向け検索テーブルの処理ステップ＃１、スレッド＃３向け検索テーブルの処理ステップ＃１、スレッド＃２向け検索テーブルの処理ステップ＃２、スレッド＃３向け検索テーブルの処理ステップ＃２、スレッド＃２向け検索テーブルの処理ステップ＃３、スレッド＃３向け検索テーブルの処理ステップ＃３、スレッド＃２向け検索テーブルの処理ステップ＃４、スレッド＃３向け検索テーブルの処理ステップ＃４の順に探す。図１７と図１８でコンテキスト＃１へのポインタがなくなっている場合は、スレッド＃１はスレッド＃２向け検索テーブルでコンテキスト＃３へのポインタを見つけ、コンテキスト＃３を利用する。
【０１３０】
上記コンテキスト検索処理においては、コンテキストがない場合、すなわち、コンテキストが０の場合、他スレッド向けのスレッド向け検索テーブルを参照して、コンテキストを探すようにしていたが、例えば、コンテキストが所定数以下である場合に、他スレッド向けのスレッド向け検索テーブルを参照して、コンテキストを探すようにしてもよい。
【０１３１】
また、上記コンテキスト検索処理のステップＳ４３によると、各スレッド間の利用可能なコンテキストの数に基づいて、スレッド間の負荷の偏りを低減することができるが、この処理以外に、後述する変形例（図３４参照）に示す負荷分散スレッドにより調整するようにしてもよい。
【０１３２】
図２８は、実施例１に係るクエリ実行プラン実行処理のフローチャートである。
【０１３３】

クエリ実行プラン実行処理は、図２６のステップＳ３９に対応する。このクエリ実行プラン実行処理は、プロセッサコアがスレッドに割当たったタスクを実行することにより実現される。なお、このクエリ実行プラン実行処理を実行する論理的な機能部が、クエリ実行プラン実行部１５１に相当する。

【０１３４】

プロセッサコアは、ＤＢページ取得処理（図２９参照）を実行する（ステップＳ５１）ことにより、ＤＢ２０６におけるページを取得する。次いで、プロセッサコアは、ページにおけるデータについて検索条件と合致するものがあるかの真偽を判定する（ステップＳ５２）。例えば、索引の上位ページであれば、上位ページ内の検索処理であり、リーフページであればリーフページの検索処理である。この結果、ページにおけるデータに、検索条件と合致するデータがない場合（ステップＳ５２で「偽」）には、プロセッサは、クエリ実行プラン実行処理を終了する。

【０１３５】

一方、検索条件に合致するデータがある場合（ステップＳ５２で「真」）には、プロセッサコアは、検索条件に合致するデータが１つであるか、２つ以上であるかを判定する（ステップＳ５３）。この結果、検索条件に合致するデータが１つである場合（ステップＳ５３で「１つ」）には、プロセッサコアは、処理をステップＳ５５に進める。一方、検索条件に合致するデータが２つ以上である場合（ステップｓ５３で「２つ以上」）には、プロセッサコアは、新規タスク追加処理（図３０参照）を実行し（ステップＳ５４）、処理をステップＳ５５に進める。

【０１３６】

ステップＳ５５では、プロセッサコアは、当該タスクによる処理ステップにおけるＤＢのページに対する処理を実行する。ここで、ＤＢのページに対する処理とは、例えば、索引の上位ページであれば検索条件に合致するページ番号を読み出す処理であり、リーフページであれば検索条件に合致するローＩＤを読み出す処理であり、表のページであればレコードのカラムを読み出す処理である。

【０１３７】

次いで、プロセッサコアは、次のＤＢのページと、当該ＤＢページに対する処理を決定し（ステップＳ５６）、ステップＳ５７に処理を進める。

【０１３８】

ステップＳ５７では、プロセッサコアは、取得しているＤＢページを開放する。次いで、ステップＳ５８では、プロセッサコアは、次の処理があるか否かを判定する。具体的には、現在行っている処理ステップが完了しており、当該処理ステップを含む処理ブロックにおいて次の処理ステップがない場合に「無」と判定する。この結果、次の処理がある場合（ステップＳ５８で「有」）には、プロセッサコアは、処理をステップＳ５１に進める一方、次の処理がない場合（ステップＳ５８で「無」）には、処理結果をクエリ実行部１４４に渡し（ステップＳ５９）、クエリ実行プラン実行処理を終了する。

【０１３９】

ここで、次のＤＢページと、当該ＤＢページに対する処理の決定について、図４～図７に示すＤＢ２０６に対して、ｃ１＝１３０を検索条件として、Ｐａｒｔ索引を索引検索する場合を例に説明する。

【０１４０】

最初に索引検索を開始している場合においては、プロセッサコアは、索引のルートページ（ページ番号「Ｐ１」のページ）を次のＤＢページと決定し、当該ページに対して「１３０」というキーを探す上位ページ内の検索処理をＤＢページに対する処理として決定し、処理を開始する。ステップＳ５１で、プロセッサコアはページＰ１を読み込み、ステップＳ５２で当該ページＰ１の中でｃ１「１３０」を含むエントリを探す。ｃ１「２００」を含むエントリを１つ見つけるので、ステップＳ５５とステップ５６で、次の処理としてページＰ３に対して上位ページ内の検索処理をＤＢページに対する処理と決定する。また、ステップ５１からステップＳ５５で、ページＰ３に対する処理を行う。プロセッサコア

は、ページＰ３を読み込み、当該ページＰ３でｃ１「１３０」を含むエントリを探し、ｃ１「１３０」を含むエントリにおいてページＰ８へのポインタを見つける。この結果、ページＰ８を次のＤＢページと決定し、当該ページＰ８に対してリーフページ内の検索処理をＤＢページに対する処理と決定する。

【０１４１】

プロセッサコアは、ステップ５１からステップＳ５３で、ページＰ８を読み込み、当該ページＰ８でｃ１「１３０」を含むエントリを探し、Ｐａｒｔ表のページ「Ｐ１００」と、スロット番号「２」を見つける。ここで条件に合致するデータが３つあるので、当該タスクで処理するデータ以外の２つのデータの処理を行うために、新規タスク追加処理（ステップ５４）を行う。本実施例では、当該タスクで処理するデータを最初のデータとし、ステップ５６でＰａｒｔ表のページＰ１００を次のＤＢページと決定し、当該ページＰ１００に対してスロット番号２にあるレコードを取得する処理をＤＢページに対する処理と決定する。

【０１４２】

図２９は、実施例１に係るＤＢページ取得処理のフローチャートである。

【０１４３】

ＤＢページ取得処理は、クエリ実行プラン実行処理（図２８）のステップＳ５１に対応する。このＤＢページ取得処理は、プロセッサコアがスレッドに割当たったタスクを実行することにより実現される。

【０１４４】

プロセッサコアは、ＤＢバッファ管理部１４７において、取得対象のＤＢページに対応するバッファページ（ＤＢバッファページ）を検索し（ステップＳ６１）、対応するＤＢバッファページの有無を確認する（ステップＳ６２）。

【０１４５】

この結果、対応するＤＢバッファページがある場合（ステップＳ６２で「有」）には、プロセッサコアは、ＤＢ２０６からの当該ページの読込みが完了しているか否かを判定し（ステップＳ６３）、読み込みが完了している場合（ステップＳ６３で「完了」）には、ＤＢページ取得処理を終了する一方、読み込みが完了していない場合（ステップＳ６３で「未完」）には、ステップＳ６６に処理を進める。

【０１４６】

一方、対応するＤＢバッファページがない場合（ステップＳ６２で「無」）には、プロセッサコアは、ＤＢバッファ管理部１４７から空きＤＢバッファページを取得し（ステップＳ６４）、ＤＢ２０６に対して対応するページを空きＤＢバッファページに読込むためのＤＢページ読込み要求を発行し（ステップＳ６５）、処理をステップＳ６６に進める。

【０１４７】

ステップＳ６６では、プロセッサコアは、ページの読込みが完了するのを待つ。ここで、プロセッサコアは、ページの読込みが完了するまで待つ方式、すなわち、同期Ｉ／Ｏを採用せずに、ページの読込みが完了していなくて他の処理を実行する方式、すなわち、非同期Ｉ／Ｏを採用しても良い。例えば、プロセッサコアは、実行中のタスクの処理を中断して待ち状態とし、タスク実行状態情報を待ちリストにつけかえる。そして、別のスレッド（又は、別タスク）により対応するページの読込みの完了を確認する。そして、当該別のスレッド（別のスレッドを実行するプロセッサコア）がページの読込みの完了を確認した場合には、当該タスクのタスク実行状態情報を実行可能リストにつけかえ、当該タスクの処理を再開させるようにしてもよい。このように、非同期Ｉ／Ｏを採用すると、プロセッサコアは、ページの読込み完了を待たずに、他のタスクの実行を行うことができるようになり、ＤＢＭＳ１４１における処理効率を向上することができる。なお、読み込みが完了した場合には、プロセッサコアは、ＤＢページ取得処理を終了する。

【０１４８】

図３０は、実施例１に係る新規タスク追加処理のフローチャートである。

【０１４９】

新規タスク追加処理は、クエリ実行プラン実行処理（図２８）のステップＳ５４に対応する。この新規タスク追加処理は、ステップＳ５３で条件に合致するデータが２以上ある場合に、合致するデータ中の１つのデータ（例えば、最初のデータ）以外のデータを対象にして実行される。

【０１５０】

プロセッサコアは、処理対象のデータに基づいて、コンテキスト８２を作成する（ステップＳ７１）。次いで、プロセッサコアは、作成したコンテキスト８２をスレッド間で共有するか否かを判定するためのコンテキスト共有判定処理（図３１参照）を実行する（ステップＳ７２）。次いで、プロセッサコアは、作成したコンテキスト８２をコンテキスト管理情報８０に登録するコンテキスト登録処理（図３２参照）を実行する（ステップＳ７３）。

【０１５１】

次いで、プロセッサコアは、新たなタスクを生成することが可能であるか否かを判定する（ステップＳ７４）。新たなタスクを生成することが可能であるか否かは、例えば、ＤＢＭＳ１４１において生成されているタスクの数が、生成可能なタスクの数の上限値に達しているか否かを判定することにより判定できる。

【０１５２】

この結果、タスクを生成可能である場合（ステップＳ７４で「可」）には、プロセッサコアは、新たなタスクを生成するためのタスク生成処理（図３３参照）を実行し（ステップＳ７５）、新規タスク追加処理を終了する。一方、タスクを生成可能でない場合（ステップＳ７４で「否」）には、タスクを生成することなく新規タスク追加処理を終了する。

【０１５３】

図３１は、実施例１に係るコンテキスト共有判定処理のフローチャートである。

【０１５４】

コンテキスト共有判定処理は、新規タスク追加処理（図３０）のステップＳ７２に対応する。このコンテキスト共有判定処理は、プロセッサコアがスレッドに割当たったタスクを実行することにより実現される。

【０１５５】

プロセッサコアは、生成したコンテキストに関する処理ステップのスレッド間共有フラグを参照する（ステップＳ８１）。その結果、処理ステップにスレッド間共有フラグが設定されている場合（ステップＳ８１で「フラグ設定あり」）には、プロセッサコアは、当該コンテキストを複数のスレッドから利用可能とするスレッド間共有と判断し（ステップＳ８２）、コンテキスト共有判定処理を終了する。一方、処理ステップにスレッド間共有フラグが設定されていない場合（ステップＳ８１で「フラグ設定なし」）には、プロセッサコアは、当該コンテキストを一つのスレッドにより利用可能とするスレッド間非共有と判断し（ステップＳ８３）、コンテキスト共有判定処理を終了する。

【０１５６】

なお、コンテキスト共有判定処理は、スレッド間共有フラグに基づいて、生成するコンテキストをスレッド間共有するかしないかを判定していたが、これに限られない。例えば、プロセッサコアが、ＤＢＭＳ１４１の実行状態に基づいて、生成するコンテキストをスレッド間共有するかしないかを判定するようにしてもよい。

【０１５７】

例えば、ＤＢＭＳ１４１の実行状態として、ＤＢＭＳ１４１の現存のタスク数を採用する。プロセッサコアは、現存のタスク数が所定数以下である場合には、生成するコンテキストをスレッド間共有と判定し、現存のタスク数が所定数以下でない場合には、生成するコンテキストをスレッド間非共有と判定しても良い。

【０１５８】

また、ＤＢＭＳの実行状態として、コンテキストに含まれる中間結果８２ｂを採用し、プロセッサコアは、前記コンテキストに含まれる中間結果８２ｂのデータ量が所定量以下である場合には、生成するコンテキストをスレッド間共有すると判定し、前記コンテキス

トに含まれる中間結果８２ｂのデータ量が所定量以下でない場合には、生成するコンテキストをスレッド間非共有と判定しても良い。

【０１５９】

図３２は、実施例１に係るコンテキスト登録処理のフローチャートである。

【０１６０】

コンテキスト登録処理は、新規タスク追加処理（図３０）のステップＳ７３に対応する。このコンテキスト登録処理は、プロセッサコアがスレッドに割当たったタスクを実行することにより実現される。

【０１６１】

プロセッサコアは、作成されたコンテキストをコンテキスト管理情報８０の管理リストに登録する（ステップＳ９１）。具体的には、プロセッサコアは、管理リストに接続された最後のコンテキストの後ろに作成されたコンテキストを接続する。

【０１６２】

次いで、プロセッサコアは、コンテキスト共有判定処理（図３１参照）の結果を確認する（ステップＳ９２）。この結果がスレッド間共有である場合（ステップＳ９２で「共有」）には、複数のスレッド向け検索テーブルに、作成されたコンテキストへのポインタを登録し、コンテキスト登録処理を終了する。本実施例では、ＤＢアクセス処理を実行する全てのスレッド向け検索テーブルにコンテキストへのポインタを登録する（ステップＳ９３）。この他に、特定のスレッド向け検索テーブルにコンテキストへのポインタを登録してもよい。

【０１６３】

特定のスレッド向け検索テーブルにコンテキストへのポインタを登録するケースを説明する。例えば、計算機のハードウェア構成情報に基づき、登録するスレッド向け検索テーブルを特定する。ハードウェア構成情報には、プロセッサ構成やキャッシュ構成やメモリ構成が考えられる。例えば、一つのプロセッサにある複数のプロセッサコアが実行している複数のスレッドにおいて、利用可能なコンテキストのタスク生成可能数の総和が最も少ない前記複数のスレッドに対応するスレッド向けのスレッド向け管理テーブルに登録する。本実施例において、スレッド＃２を実行しているプロセッサコアとスレッド＃３を実行しているプロセッサコアが同じプロセッサであり、前記プロセッサはスレッド＃１を実行しているプロセッサと異なり、コンテキスト＃１へのポインタがスレッド＃２のスレッド向け検索テーブルとスレッド＃３のスレッド向け検索テーブルに登録されている状況を考える。処理ステップ＃１に対する新しいコンテキスト＃４が生成された場合には、コンテキストが少ないプロセッサのプロセッサコアが実行しているスレッドスレッド向け検索テーブルに登録する。このケースでは、スレッド＃１のスレッド向け検索テーブルに登録する。この例では、１つのスレッド向け検索テーブルに登録したが、スレッド＃１を実行するプロセッサコアのプロセッサが他にＤＢアクセス処理を実行するスレッドを実行している場合は、複数のスレッド向け検索テーブルに登録することになる。

【０１６４】

このほか、コンテキストを生成したプロセッサコアを含むプロセッサに対応する複数のスレッドに対応するスレッド向けのスレッド向け管理テーブルに登録してもよい。また、一つのプロセッサにある複数のプロセッサコアが実行している複数のスレッドにおいて、利用可能なコンテキストの総数が少ない前記複数のスレッドに対応するスレッド向けのスレッド向け管理テーブルに登録してもよい。また、プロセッサ内でキャッシュを共有しているプロセッサコアが実行している複数のスレッドのスレッド向け管理テーブルに登録してもよい。また、コンテキストを記録するメモリに近いプロセッサコアが実行している複数のスレッドのスレッド向け管理テーブルに登録してもよい。また、コンテキストを利用する際に参照するＤＢバッファページが記録されているメモリに近いプロセッサコアが実行している複数のスレッドのスレッド向け管理テーブルに登録してもよい。

【０１６５】

一方、この結果がスレッド間非共有である場合（ステップＳ９２で「非共有」）には、

プロセッサコアは、１つのスレッド向け検索テーブルにポインタを登録し、コンテキスト登録処理を終了する。本実施例では、自身が実行しているスレッド（自スレッド）向けのスレッド向け検索テーブルに、作成されたコンテキストへのポインタを登録し（ステップＳ９４）、コンテキスト登録処理を終了する。このほかに、利用可能なコンテキストが最も少ないスレッド向け検索テーブルに登録してもよく、また、利用可能なコンテキストのタスク生成可能数の総和が最も少ないスレッド向け検索テーブルに登録してもよい。

【０１６６】

図３３は、実施例１に係るタスク生成処理のフローチャートである。

【０１６７】

タスク生成処理は、新規タスク追加処理（図３０）のステップＳ７５に対応する。このタスク生成処理は、プロセッサコアがスレッドに割当たったタスクを実行することにより実現される。

【０１６８】

次いで、プロセッサコアは、コンテキスト共有判定処理（図３１参照）の結果を確認する（ステップＳ１０１）。この結果がスレッド間共有である場合（ステップＳ１０１で「スレッド間共有」）には、プロセッサコアは、タスクを生成し、コンテキストのポインタを登録したスレッド向け検索テーブルに対応する２つ以上のスレッドにタスクを割り当てる（ステップＳ１０２）。生成するタスクの合計はコンテキストの生成可能数を上限とする。各スレッドに割り当てるタスクの数は、コンテキストの生成可能数をスレッド数で割った数とする。その後、プロセッサコアは、タスク生成処理を終了する。

【０１６９】

一方、この結果がスレッド間非共有である場合（ステップＳ１０１で「スレッド間非共有」）には、プロセッサコアは、タスクを生成し、コンテキストのポインタを登録したスレッド向け検索テーブルに対応する１つのスレッドにタスクを割り当てる（ステップＳ１０３）。生成するタスクの数はコンテキストの生成可能数を上限とする。ここで、タスクを割り当てるスレッドとしては、プロセッサコアが実行している自スレッドであっても良く、自スレッド以外のスレッドであっても良い。ステップＳ１０３の後、プロセッサコアは、タスク生成処理を終了する。

【０１７０】

次に、本実施例の変形例について説明する。

【０１７１】

上記実施例において、クエリ実行部１４４が以下に示す負荷分散処理を実行するようにしてもよい。

【０１７２】

図３４は、変形例に係る負荷分散処理のフローチャートである。

【０１７３】

負荷分散処理は、クエリ実行部１４４により実行されるが、具体的には、プロセッサコアが、ＤＢ処理を行うためのスレッド以外のスレッド（負荷分散スレッド）を実行することにより実現することができる。この負荷分散処理は、クライアント通信制御部１４２がクエリを受け付けた後に開始される。

【０１７４】

プロセッサコアは、クエリ処理が終了したか否かを判定し（ステップＳ１１１）、クエリ処理が終了した場合（ステップＳ１１１で「終了」）には、負荷分散処理を終了する。

【０１７５】

一方、クエリ処理が終了していない場合（ステップＳ１１１で「未終了」）には、プロセッサコアは、各スレッド向け検索テーブルから利用可能なコンテキストにおけるタスク生成可能数の総和を計算する（ステップＳ１１２）。

【０１７６】

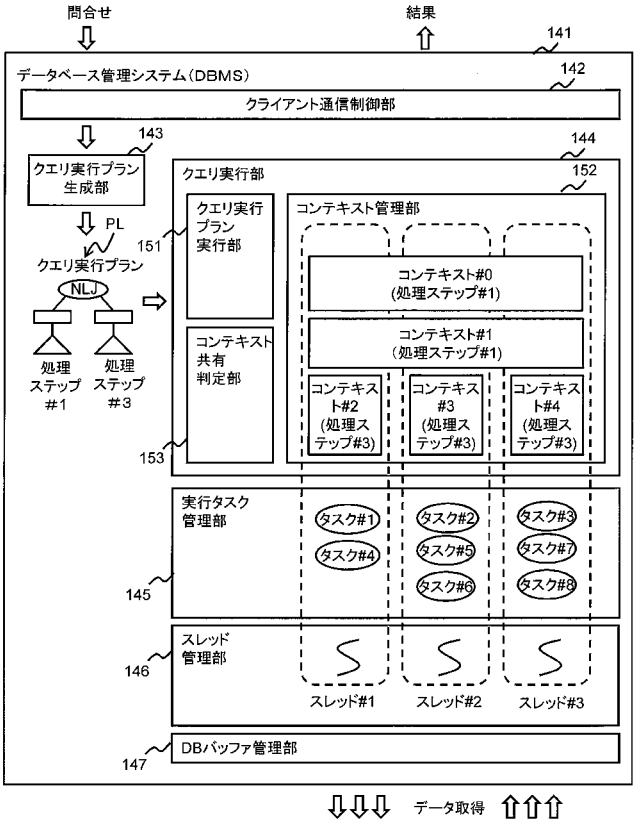
次いで、プロセッサコアは、各スレッドが生成可能なタスク生成可能数の総和に偏りがあるか否かを判定する（ステップＳ１１３）。ここで、プロセッサコアは、例えば、タス

ク生成可能数が所定数（例えば、０）以下である場合に偏りがあると判定しても良い。
【０１７７】
この結果、各スレッドが生成可能なタスク生成可能数の総和に偏りがない場合（ステップＳ１１３で「無」）には、処理をステップＳ１１５に進める。
【０１７８】
一方、各スレッドで生成可能なタスク生成可能数の総和に偏りがある場合（ステップＳ１１３で「有」）には、プロセッサコアは、コンテキストの位置を変更、すなわち、コンテキストを参照するポインタが格納されるスレッド向け検索テーブルを別のスレッド向け検索テーブルに変更することにより、各スレッドで生成可能なタスク生成可能数の総和の偏りを減らすようにする。具体的には、タスク生成可能数の総和が最大のスレッド向けのスレッド向け検索テーブルにより利用可能なコンテキストのポインタを、タスク生成可能数の総和が少ないスレッド向けのスレッド向け検索テーブルに登録する。この後、プロセッサコアは、処理をステップＳ１１５に進める。
【０１７９】
ステップＳ１１５では、プロセッサコアは、当該負荷分散処理を所定時間スリープし、処理をステップＳ１１１に進める。
【０１８０】
この負荷分散処理により、各スレッドに対する負荷を適切に分散させることができる。
【０１８１】
なお、上記負荷分散処理では、各スレッドで生成可能なタスク生成可能数の総和の偏りに基づいて、コンテキストを利用するスレッドの変更を行うようにしていたが、タスク生成可能数の総和の偏りと異なる、スレッドに関わる実行状態に基づいて、スレッドの負荷を把握し、コンテキストを利用するスレッドを変更するようにしてもよい。例えば、各スレッドにおけるコスト計算を行い、前記コストに基づいて、コンテキストを利用するスレッドを変更するようにしても良い。コスト計算の例として次のような値が考えられる。コンテキスのコストを当該コンテキストから処理する処理ステップ数と生成可能数の積とし、当該スレッドから利用可能なコンテキストのコストの総和を当該スレッドのコストとする。
【０１８２】
なお、上記負荷分散処理は、ＤＢ処理を行うためのスレッドが実行してもよい。例えば、スレッドの終了時（ステップＳ３２の「無」と判定した場合）や、タスクの終了時（ステップＳ３７で「なし」と判定した場合）に実行してもよい。この場合、負荷分散処理は、ステップＳ１１２からステップＳ１１４までを実行する。
【実施例２】
【０１８３】
以下、実施例２を説明する。その際、実施例１との相違点を主に説明し、実施例１との共通点については説明を省略或いは簡略する。
【０１８４】
図３５は、実施例２に係る計算機システムの構成を示す。
【０１８５】
アプリケーションサーバ（以下、ＡＰサーバ）３５０２は、ＤＢＭＳ１４１が動作する計算機（以下、ＤＢサーバ）１００に、通信ネットワーク３５１２を介して通信できるように接続されている。また、ＤＢサーバ１００は、外部ストレージ装置２００に、通信ネットワーク３００を介して通信できるように接続されている。ユーザ端末（クライアント端末）３５０１は、ＡＰサーバ３５０２に、通信ネットアーク３５１１を介して通信できるように接続されている。ＤＢサーバ１００は、ＤＢ２０６を管理するＤＢＭＳ１４１を実行する。外部ストレージ装置２００は、ＤＢ２０６を格納する。ＡＰサーバ３５０２は、ＤＢサーバ１００で実行されるＤＢＭＳ１４１に対してクエリを発行するＡＰを実行する。ユーザ端末３５０１は、ＡＰサーバ３５０２で実行されるＡＰに要求を出す。なお、ユーザ端末３５０１、又は、ＡＰサーバ３５０２は、複数存在しても良い。

【０１８６】
ＡＰサーバ管理端末３５０３は、通信ネットワーク３５１４を介してＡＰサーバ３５０２に接続されている。ＤＢサーバ管理端末３５０４は、通信ネットワーク３５１５を介してＤＢサーバ１００に接続されている。ストレージ管理端末３５０５は、通信ネットワーク３５１６を介して外部ストレージ装置２００に接続されている。ＡＰサーバ管理端末３５０３は、ＡＰサーバ３５０２を管理する端末である。ＤＢサーバ管理端末３５０４は、ＤＢサーバ１００を管理する端末である。ストレージ管理端末３５０５は、外部ストレージ装置２００を管理する端末である。ＤＢサーバ管理者又はユーザは、ＤＢサーバ管理端末３５０４から、ＤＢＭＳ１４１に関する設定を行っても良い。なお、管理端末３５０３～３５０５のうちの少なくとも二つが共通（一体）であっても良い。また、通信ネットワーク３５１１、３５１２、３５１４、３５１５、３５１６、及び３００のうちの少なくとも二つが共通（一体）であっても良い。
【０１８７】
実施例２では、例えば、下記の通り処理が実行される。
（Ｓ１２１）ユーザ端末３５０１は、ＡＰサーバ３５０２に要求（以下、ユーザ要求）を発行する。
（Ｓ１２２）ＡＰサーバ３５０２は、Ｓ１２１で受信したユーザ要求に従いクエリを生成する。そして、生成したクエリをＤＢサーバ１００に発行する。
（Ｓ１２３）ＤＢサーバ１００は、ＡＰサーバ３５０２からのクエリを受け付け、受け付けたクエリを実行する。ＤＢサーバ１００は、受け付けたクエリの実行において必要なデータ入出力要求（例えばデータ読出し要求）を外部ストレージ装置２００に発行する。ＤＢサーバ１００は、一つのクエリの実行において、複数のデータ入出力要求を並行して発行することがある。そのため、ＤＢサーバ１００は、一つのクエリの実行において、Ｓ１２３の要求を複数回並行して行うことがある。
（Ｓ１２４）外部ストレージ装置２００は、Ｓ１２３で発行されたデータ入出力要求について、ＤＢサーバ１００に応答する。外部ストレージ装置２００は、Ｓ１２４の応答を複数回並行して行うことがある。
（Ｓ１２５）ＤＢサーバ１００は、クエリの実行結果を生成し、ＡＰサーバ３５０２に送信する。
（Ｓ１２６）ＡＰサーバ３５０２は、クエリの実行結果を受信する。そして、該実行結果に従う、Ｓ１２１で受信したユーザ要求に対する回答を、ユーザ端末３５０１に送信する。
【０１８８】
なお、ＡＰサーバ３５０２に発行されるユーザ要求、又は、ＤＢサーバへ発行されるクエリは、同時に複数あっても良い。
【０１８９】
以上、実施例に基づいて説明したが、本発明は上述した実施例に限られず、他の様々な態様に適用可能である。
【符号の説明】
【０１９０】
１４１…データベース管理システム（ＤＢＭＳ）

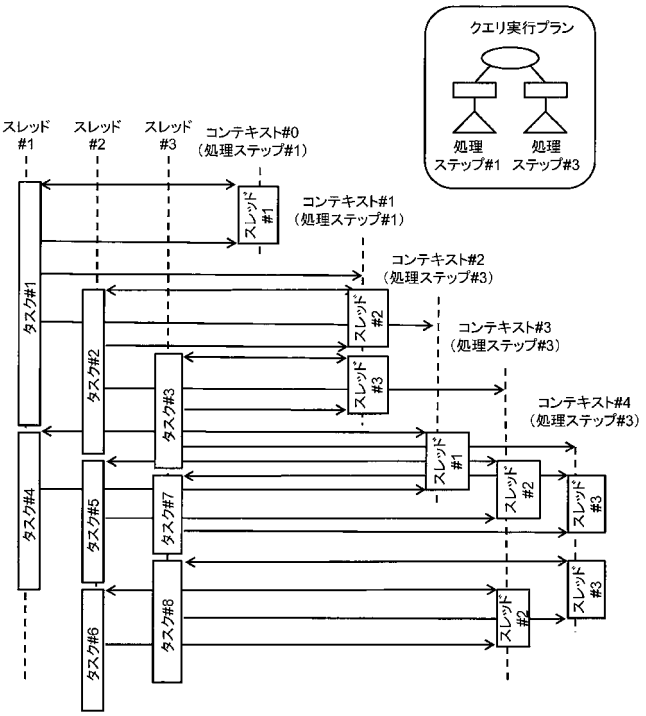
【図１】

Fig. 1



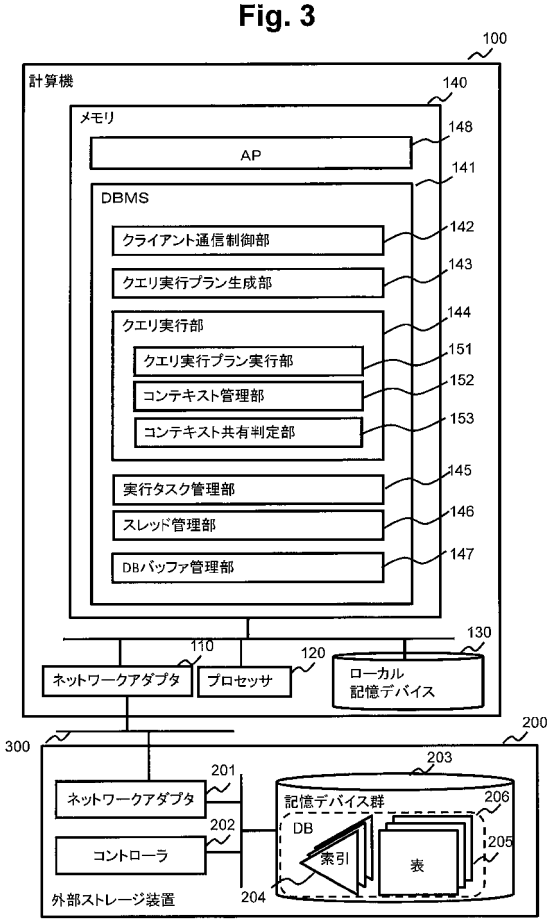
【図２】

Fig. 2



【図３】

Fig. 3



【図４】

Fig. 4

表定義、索引定義

Part表:
CREATE TABLE PART (c1 INTEGER, c2 INTEGER)
Part索引:
CREATE INDEX IDX_PART ON PART (c1)
Lineitem表:
CREATE TABLE LINEITEM (c3 INTEGER, c4 CHAR(10))
Lineitem索引:
CREATE INDEX IDX_LINEITEM ON LINEITEM (c3)

【図５】

Fig. 5

Part表の論理イメージ

c1	c2
10	id11
⋮	
130	id131
130	id132
130	id133
⋮	
300	id301

【図６】

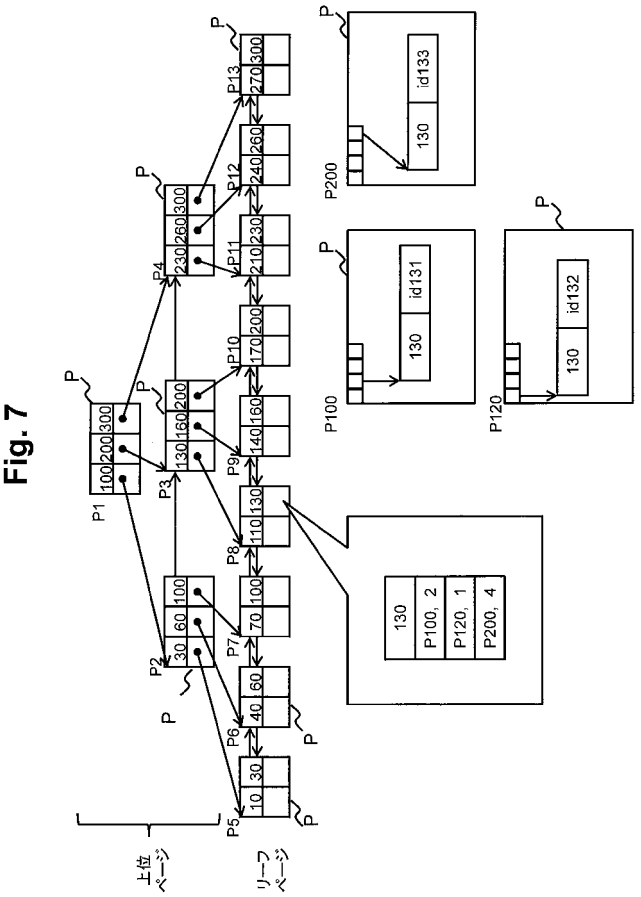
Fig. 6

Lineitem表の論理イメージ

c3	c4
id11	A
⋮	
id131	D
id131	E
id132	A
id132	B
id132	C
id133	F
id133	G
⋮	
id301	Z

【図７】

Fig. 7



【図８】

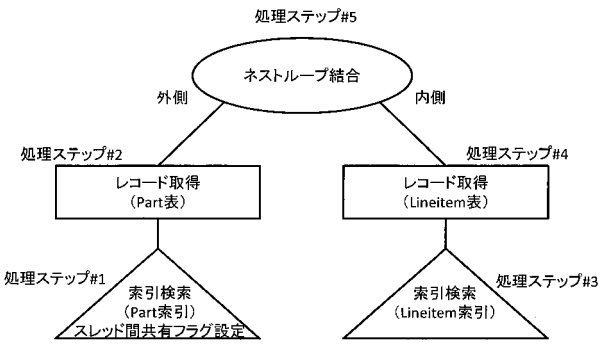
Fig. 8

クエリ

```
SELECT    c1, c4
FROM      part, lineitem
WHERE     c1 = 130   and c2 = c3
```

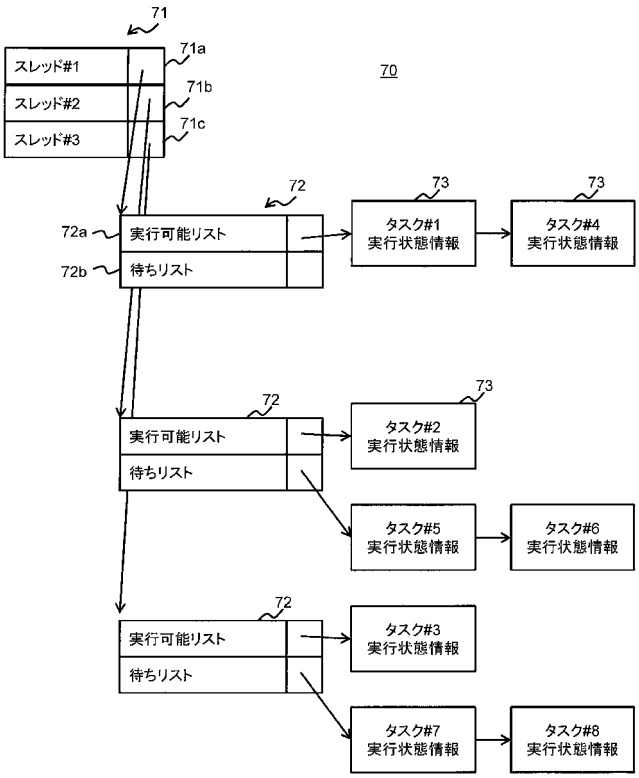
【図９】

Fig. 9



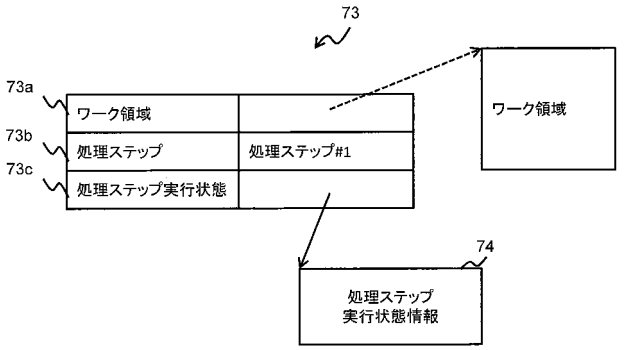
【図１０】

Fig. 10



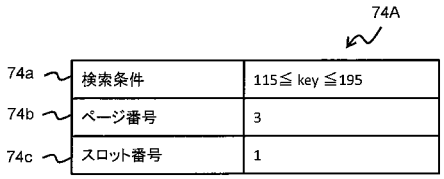
【図１１】

Fig. 11



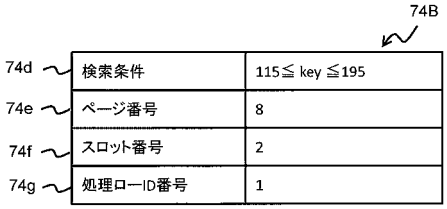
【図１２】

Fig. 12



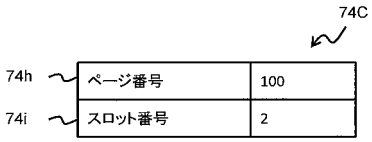
【図１３】

Fig. 13



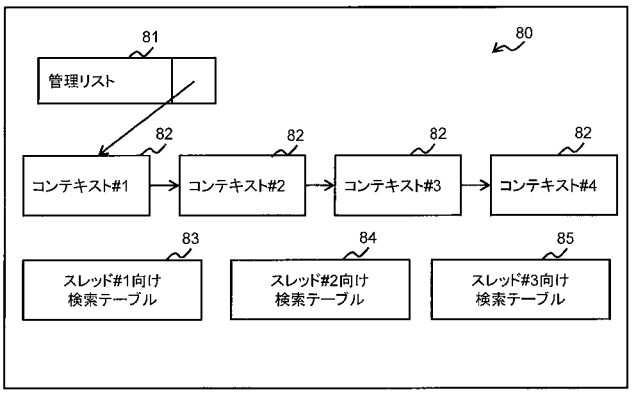
【図１４】

Fig. 14



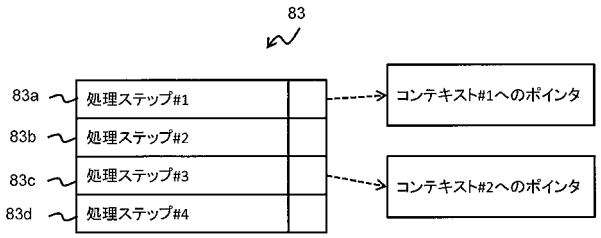
【図１５】

Fig. 15



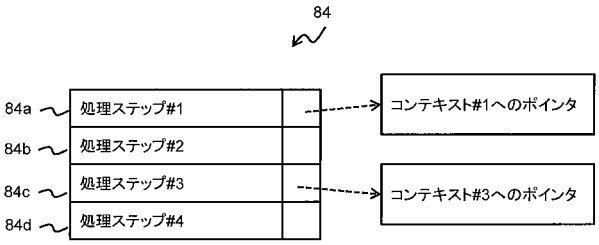
【図１６】

Fig. 16



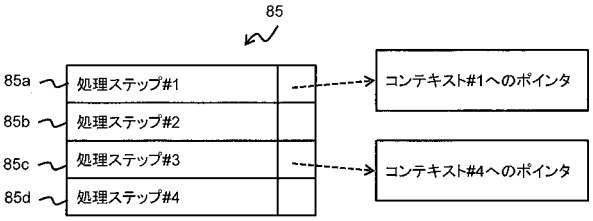
【図１７】

Fig. 17



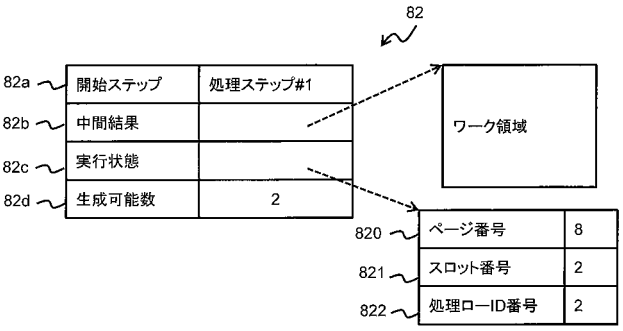
【図１８】

Fig. 18



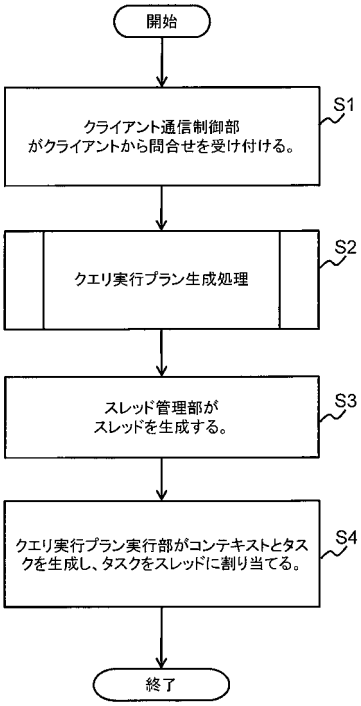
【図１９】

Fig. 19



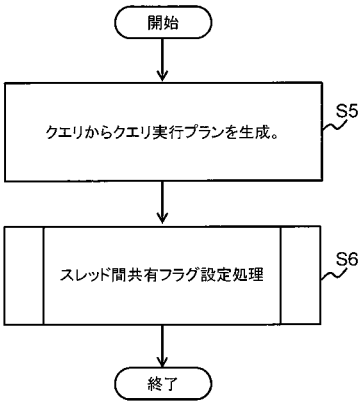
【図２０】

Fig. 20



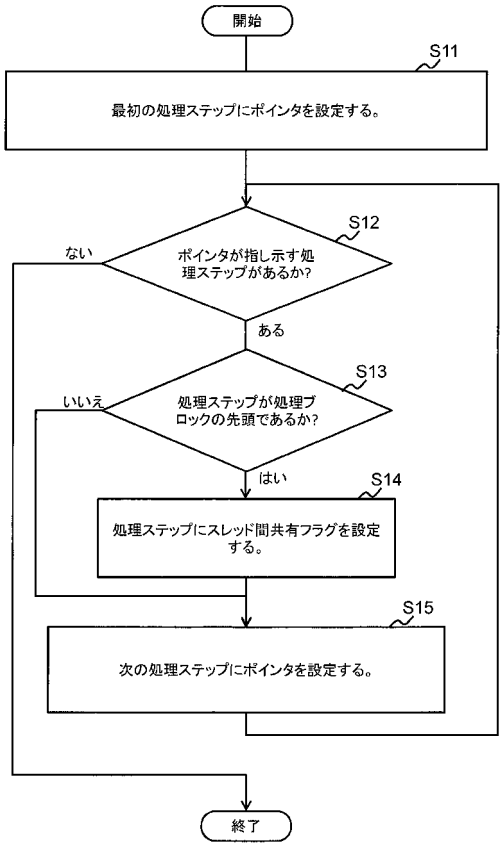
【図２１】

Fig. 21



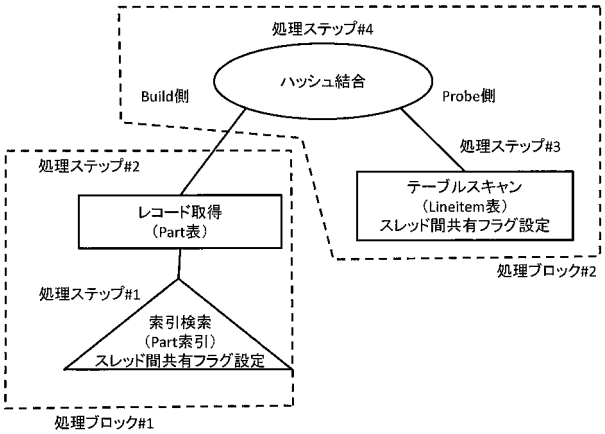
【図２２】

Fig. 22



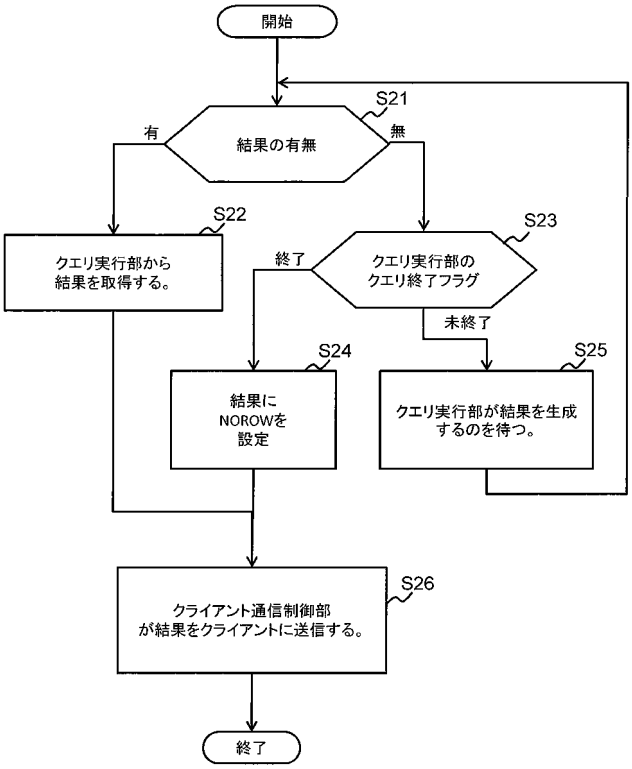
【図２３】

Fig. 23



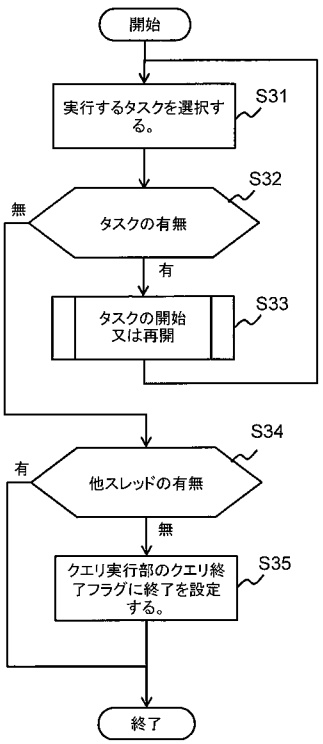
【図２４】

Fig. 24



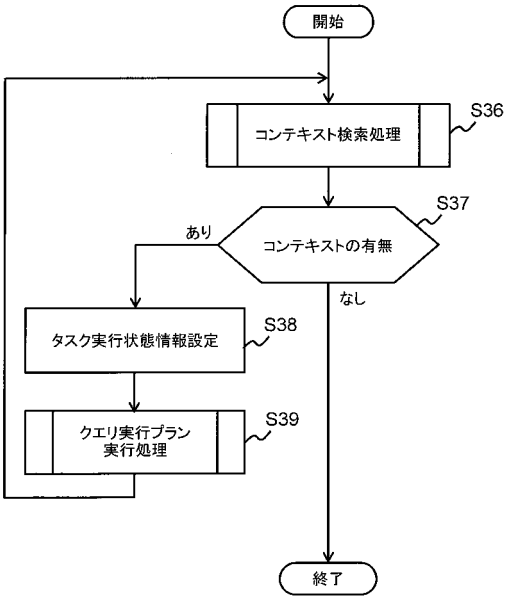
【図２５】

Fig. 25



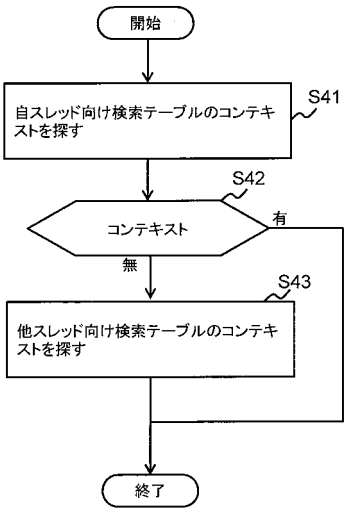
【図２６】

Fig. 26



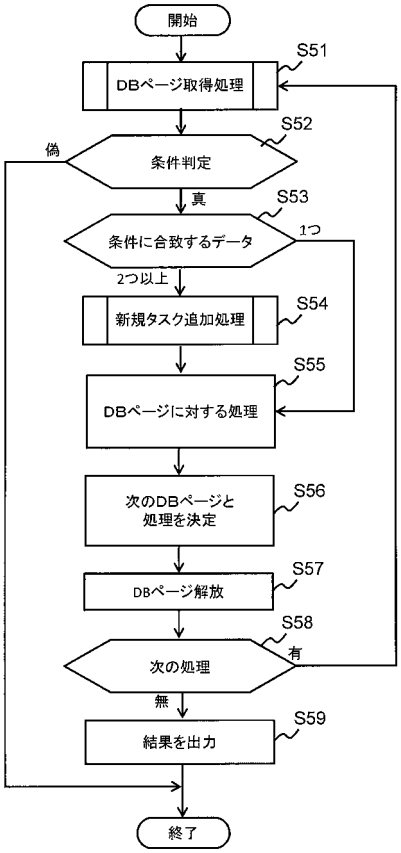
【図２７】

Fig. 27



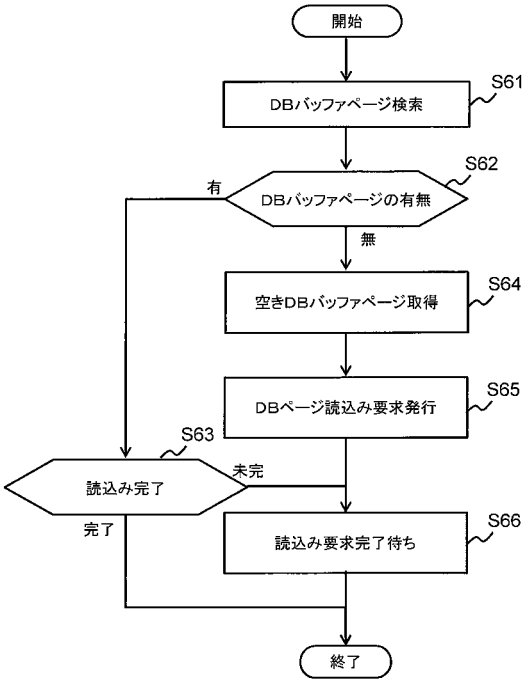
【図２８】

Fig. 28



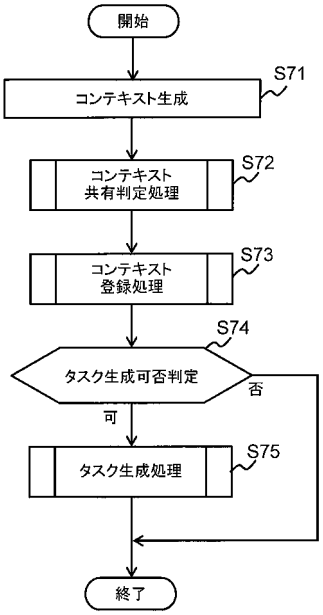
【図２９】

Fig. 29



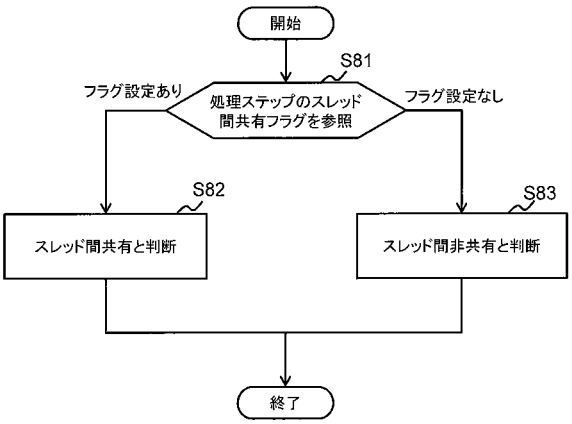
【図３０】

Fig. 30



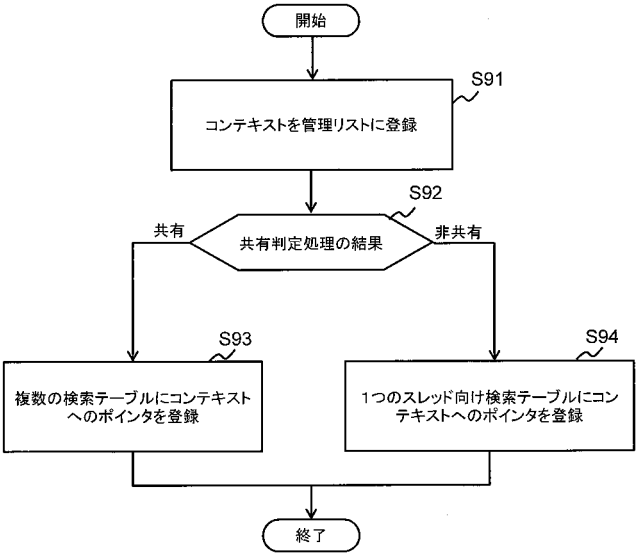
【図３１】

Fig. 31



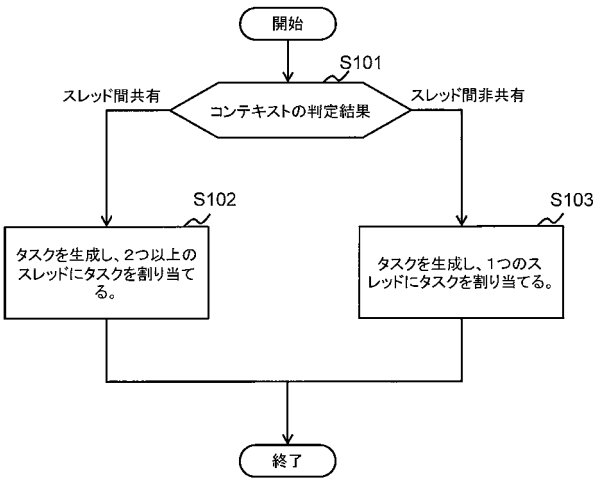
【図３２】

Fig. 32



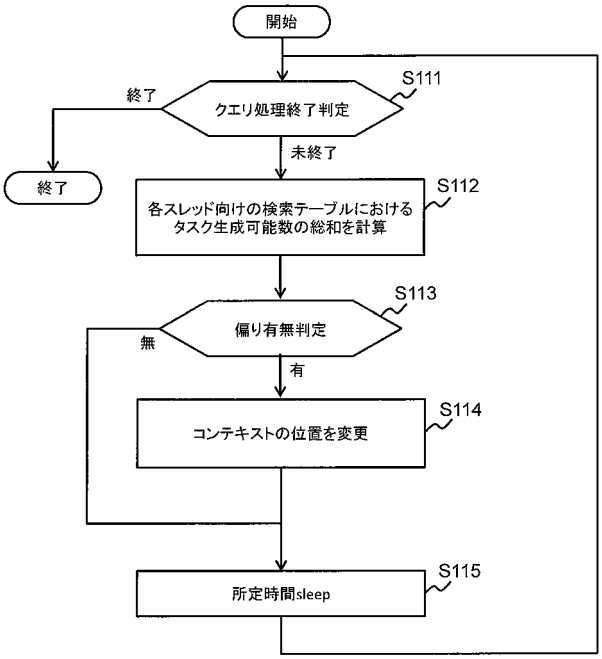
【図３３】

Fig. 33



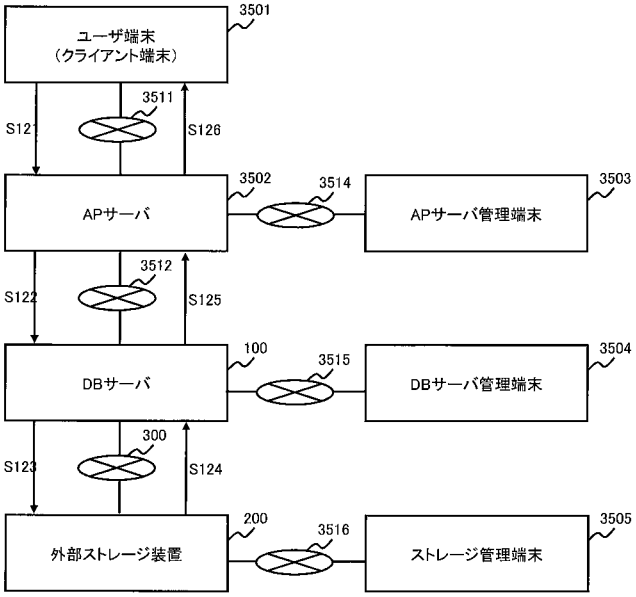
【図３４】

Fig. 34



【図３５】

Fig. 35



【手続補正書】
【提出日】平成27年12月17日(2015.12.17)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
プロセッサコアを有する計算機により実現されデータベースを管理するデータベース管理システムであって、
前記データベースへのクエリを受け付けるクエリ受付部と、
前記受け付けたクエリを実行するために必要な処理ステップと前記処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成するクエリ実行プラン生成部と、
前記生成したクエリ実行プランに基づいて前記受け付けたクエリを実行し、前記受け付けたクエリの実行において、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行するクエリ実行部と
を有し、
前記クエリ実行部は、
前記受け付けたクエリの実行において生成されたタスクの実行状態を表す情報を含むコンテキストを生成し、前記生成したコンテキストを、前記生成されたタスクが割り当てられたスレッドと対応付けて管理し、
前記受け付けたクエリの実行において、複数のプロセッサコアのうちの２以上のプロセッサコアの各々にスレッドを割り当て、各々のスレッドにおいて、そのスレッドに対応付けられている前記コンテキストに基づいて、当該スレッドに割り当てられている１以上

のタスクを実行する
データベース管理システム。
【請求項２】
前記クエリ実行プラン生成部は、各前記処理ステップに関わるコンテキストを、複数のスレッド間で共有するか、複数のスレッド間で共有しないかを判定し、
前記クエリ実行部は、前記判定結果に基づいて前記コンテキストを管理する
請求項１に記載のデータベース管理システム。
【請求項３】
前記クエリ実行プラン生成部は、前記処理ステップの前記クエリ実行プランにおける他の処理ステップとの先行又は後続関係に基づいて、前記処理ステップに関わるコンテキストを複数のスレッド間で共有するか、又は共有しないかを判定する
請求項２に記載のデータベース管理システム。
【請求項４】
前記プロセッサコアにより実行されるスレッドに割当たったタスクにおいて、前記データベースに対して非同期Ｉ／Ｏによるデータ読み込み要求を発行し、
前記スレッドを実行するプロセッサコアは、前記タスクにおけるデータ読み込み要求発行後に、前記データ読み込み要求に対応するデータの読み込みが完了する前に、実行可能な他のタスクの実行を行い、
前記スレッドを実行するプロセッサコアは、前記タスクにおける前記データ読み込み要求に対応するデータの読み込みが完了した後に、前記タスクの実行を再開する
請求項２又は３に記載のデータベース管理システム。
【請求項５】
前記処理ステップを実行するためのタスクを割り当て可能な前記スレッドは、前記プロセッサコアと同数であり、各スレッドを実行するプロセッサコアは、それぞれ別のプロセッサコアに設定されている
請求項２乃至４のうちのいずれか１項に記載のデータベース管理システム。
【請求項６】
前記クエリ実行プラン生成部は、前記クエリ実行プランに、並行して実行可能な処理ステップを含む複数の処理ブロックが含まれる場合に、前記処理ブロックの先頭の処理ステップに関わるコンテキストを、複数のスレッド間で共有すると判定し、前記処理ブロックの他の処理ステップに関わるコンテキストを、スレッド間で共有しないと判定する
請求項２乃至５のうちのいずれか１項に記載のデータベース管理システム。
【請求項７】
前記クエリ実行プラン生成部は、前記クエリ実行プランに、並行して実行可能な処理ステップを含む複数の処理ブロックが含まれる場合に、前記処理ブロックにおける後続の処理ステップの数が所定数以上ある処理ステップに関わるコンテキストを、複数のスレッド間で共有すると判定し、後続の処理ステップの数が所定数未満の処理ステップに関わるコンテキストを、スレッド間で共有しないと判定する
請求項２乃至６のうちのいずれか１項に記載のデータベース管理システム。
【請求項８】
前記クエリ実行部は、一つのスレッドの利用可能なコンテキストの数が所定数を下回った場合に、他のスレッドの利用可能なコンテキストを、前記一つのスレッドに割当たったタスクで利用する
請求項２乃至７のうちのいずれか１項に記載のデータベース管理システム。
【請求項９】
前記クエリ実行部は、前記複数のスレッドに関わる実行状態が所定の状態となった場合に、スレッド間の利用可能なコンテキストの数の差が所定数より小さくなるように、一つのスレッドの利用可能なコンテキストを、他のスレッドの利用可能なコンテキストに変更する
請求項２乃至８のうちのいずれか１項に記載のデータベース管理システム。

【請求項１０】
前記複数のスレッドの実行状態が所定の状態とは、前記スレッド間の利用可能なコンテキストの数の差が所定数以上となった状態である
請求項９に記載のデータベース管理システム。
【請求項１１】
前記クエリ実行部は、データベース管理システムの実行状態に基づいて、コンテキストを複数のスレッドで共有とするか否かを判定し、
前記コンテキストを複数のスレッド間で共有すると判定されている場合には、前記コンテキストを複数のスレッド間で共有するようにし、前記コンテキストを複数のスレッド間で共有しないと判定されている場合には、前記コンテキストを一つのスレッドが利用するようにする
請求項１乃至１０のうちのいずれか１項に記載のデータベース管理システム
【請求項１２】
前記データベース管理システムの実行状態とは、前記データベース管理システムにおいて現存するタスク数であり、
前記クエリ実行部は、現存する前記タスク数が所定数以下である場合には、前記コンテキストを複数のスレッドで共有とすると判定し、現存する前記タスク数が所定数以下でない場合には、前記コンテキストを共有しないと判定する
請求項１１に記載のデータベース管理システム。
【請求項１３】
前記データベース管理システムの実行状態とは、コンテキストに含まれる所定の情報であり、
前記クエリ実行部は、前記コンテキストに含まれる前記所定の情報のデータ量が所定量以下である場合には、前記コンテキストを複数のスレッドで共有とすると判定し、前記コンテキストに含まれる前記所定の情報のデータ量が所定量以下でない場合には、前記コンテキストを共有しないと判定する
請求項１２に記載のデータベース管理システム。
【請求項１４】
前記クエリ実行部は、前記コンテキストをいずれか一つのスレッドにより利用可能とし、
前記クエリ実行部は、前記スレッド間の利用可能なコンテキストの数の差が所定数以上となった場合に、前記スレッド間の利用可能なコンテキストの数の差が所定数より小さくなるように、一つのスレッドの利用可能なコンテキストを、他のスレッドの利用可能なコンテキストに変更する
請求項１乃至１３のうちのいずれか１項に記載のデータベース管理システム。
【請求項１５】
記憶資源と、
前記記憶資源に接続され１以上のプロセッサコアを有する１以上のプロセッサを含んだ制御デバイスを有し、
前記制御デバイスが、
前記データベースへのクエリを受け付け、
前記受け付けたクエリを実行するために必要な処理ステップと前記処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成し、
前記生成したクエリ実行プランに基づいて前記受け付けたクエリを実行し、前記受け付けたクエリの実行において、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行する、
ようになっており、
前記制御デバイスは、
前記受け付けたクエリの実行において生成されたタスクの実行状態を表す情報を含むコンテキストを生成し、前記生成したコンテキストを、前記生成されたタスクが割り当て

られたスレッドと対応付けて管理し、
　前記受け付けたクエリの実行において、複数のプロセッサコアのうちの２以上のプロセッサコアの各々にスレッドを割り当て、各々のスレッドにおいて、そのスレッドに対応付けられている前記コンテキストに基づいて、当該スレッドに割り当てられている１以上のタスクを実行する
計算機。
【請求項１６】
　データベースを管理するデータベース管理方法であって、
　（ａ）前記データベースへのクエリを受け付け、
　（ｂ）前記受け付けたクエリを実行するために必要な複数の１以上の処理ステップと、前記１以上の処理ステップの実行手順とを表す情報を含んだクエリ実行プランを生成し、
　（ｃ）前記生成したクエリ実行プランに基づいて、処理ステップを実行するためのタスクを動的に生成し、前記動的に生成されたタスクを実行することで、前記受け付けたクエリを実行し、
　（ｃ）において、
　前記受け付けたクエリの実行において生成されたタスクの実行状態を表す情報を含むコンテキストを生成し、前記生成したコンテキストを、前記生成されたタスクが割り当てられたスレッドと対応付けて管理し、
　前記受け付けたクエリの実行において、複数のプロセッサコアのうちの２以上のプロセッサコアの各々にスレッドを割り当て、各々のスレッドにおいて、そのスレッドに対応付けられている前記コンテキストに基づいて、当該スレッドに割り当てられている１以上のタスクを実行する
データベース管理方法。

フロントページの続き

(72)発明者　吉田　美智子
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　茂木　和彦
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　藤原　真二
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　河村　信男
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　喜連川　優
東京都文京区本郷七丁目３番１号　国立大学法人東京大学内
(72)発明者　合田　和生
東京都文京区本郷七丁目３番１号　国立大学法人東京大学内