(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2011-39820
(P2011-39820A)
(43)公開日　平成23年2月24日(2011.2.24)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/30	(2006.01)	G06F	17/30	110C	5B075
		G06F	17/30	340D	
		G06F	17/30	180D	

審査請求 未請求　請求項の数 14　ＯＬ　（全 30 頁）

(21)出願番号　特願2009-187137 (P2009-187137)
(22)出願日　平成21年8月12日 (2009.8.12)

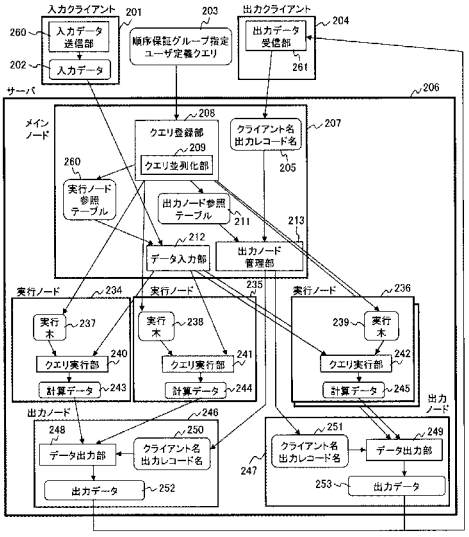
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　110000350
ポレール特許業務法人
(72)発明者　勝沼　聡
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　今木　常之
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内
(72)発明者　藤原　真二
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

Ｆターム(参考) 5B075 KK02　QR00　UU40

(54)【発明の名称】ストリームデータ処理方法及び装置

(57)【要約】
【課題】オペレータ処理後に、一つのノードで入力時に付加したタイムスタンプに従って並び替え出力して入力された順序を保つと、スループットは低下し、レイテンシは増大する。
【解決手段】ユーザ定義クエリ２０３によりユーザはグループを指定する。そして、同じグループのデータ間においては入力された順序を保証する。同じグループに属するデータを同一のノード２４６で出力することで、必要部分に対して入力された順序を保証する。異なるグループに属するデータを別のノード２４７で出力することで、複数ノードでの処理を実現し性能低下を回避する。
【選択図】図２

図 2



【特許請求の範囲】
【請求項１】
複数のノードにより、入力されるデータに対する処理を行うストリームデータ処理方法であって、
第一のノードでクエリの登録を行い、第二のノードの集合でタイムスタンプが付加されたデータをオペレータ処理し、第三のノードの集合でオペレータ処理されたデータを前記タイムスタンプに従って並び替えて出力し、
前記第一のノードは、登録した前記クエリから、入力された順序を同じグループのデータで保証するというグループが示された第一の値式の集合を抽出し、オペレータ処理を複数の前記第二のノードでデータごとに分割する方法を示す第二の値式の集合を算出し、
前記第一及び第二の値式の集合から、出力処理を複数の前記第三のノードにデータを分けて行う方法が示された第三の値式の集合を算出し、前記第三の値式の集合から前記第三のノードの集合にデータの振分ける方法を決定し、
前記第三のノードの集合における振分け方法と前記第一の値式の集合から、同じ前記第二のノードで処理されたデータが同じ前記第三のノードで処理されるようにデータを振り分ける方法を決定する、
ことを特徴とするストリームデータ処理方法。
【請求項２】
請求項１記載のストリームデータ処理方法であって、
前記第一及び第二の値式の集合における要素として共に任意の値式 Ｆがある場合に、前記値式 Ｆを第三の値式の集合における要素とする、
ことを特徴とするストリームデータ処理方法。
【請求項３】
請求項１記載のストリームデータ処理方法であって、
第一の値式の集合の要素である任意の値式 Ｆ、第二の値式の集合の要素である任意の値式 Ｇに対して、Ｆの値が同じストリームデータは必ずＧの値が同一になることが示されている場合に、第三の値式の集合における要素をＧとする、
ことを特徴とするストリームデータ処理方法。
【請求項４】
請求項１記載のストリームデータ処理方法であって、
前記第三の値式の集合と、前記第二のノードの数、前記第三のノードの数、ストリームデータに関する情報から、第三のノードの集合にデータを振分ける方法を決定し、
前記第三のノードの集合における振分け方法と前記第一の値式の集合、前記第二のノードの数、ストリームデータに関する情報から、第二のノードの集合のデータの振り分け方を、第三のノードの集合の振分け方と同様の方法または、第三のノードの集合より多くのノードにデータを振り分ける方法を解析し決定する、
ことを特徴とするストリームデータ処理方法。
【請求項５】
請求項１に記載のストリームデータ処理方法であって、
前記第一のノードまたはクライアントのノードは、前記第三のノードの集合のデータの振分け方法に関する情報を参照し、前記第三のノードの集合を検索し、前記第三のノードの集合にデータとクライアントに関する情報を送り、
前記第三のノードの集合は、データと前記クライアントの情報に基づいて出力データを前記クライアントに転送する、
ことを特徴とするストリームデータ処理方法。
【請求項６】
請求項１に記載のストリームデータ処理方法であって、
前記第一のノードは、前記クライアントから、出力するデータ間で許容する入力された順序の時刻ずれ幅が示された場合に、異なる前記第三のノードの集合から出力されるデータに対して、前記第二のノードの集合または前記第三のノードの集合は、前記第三のノード

の集合から出力されるデータのタイムスタンプ値及び許容する時刻ずれ幅を用いて出力時刻を調整し、入力されたデータの順序のずれを示された範囲に抑制する、
ことを特徴とするストリームデータ処理方法。
【請求項７】
請求項１記載のストリームデータ処理方法であって、
前記第一ノードは、複数のタイムスタンプ付加ノードを有し、同じ前記第三ノードで処理をするデータは必ず同じ前記タイムスタンプ付加ノードでタイムスタンプ付加処理をするように、複数の前記タイムスタンプ付加ノードにデータを振分ける、
ことを特徴とするストリームデータ処理方法。
【請求項８】
複数のノードにより、入力されるデータに対する処理を行うストリームデータ処理装置であって、
第一のノードでクエリの登録を行い、第二のノードの集合でタイムスタンプが付加されたデータをオペレータ処理し、第三のノードの集合でオペレータ処理されたデータを前記タイムスタンプに従って並び替えて出力し、
前記第一のノードは、登録した前記クエリから、入力された順序を同じグループのデータで保証するというグループが示された第一の値式の集合を抽出し、オペレータ処理を複数の前記第二のノードでデータごとに分割する方法を示す第二の値式の集合を算出し、
前記第一及び第二の値式の集合から、出力処理を複数の前記第三のノードにデータを分けて行う方法が示された第三の値式の集合を算出し、前記第三の値式の集合から前記第三のノードの集合にデータの振分ける方法を示す第一のテーブルを生成し、
前記第三のノードの集合における振分け方法と前記第一の値式の集合から、同じ前記第二のノードで処理されたデータが同じ前記第三のノードで処理されるようにデータを振り分ける方法を示す第二のテーブルを生成する、
ことを特徴とするストリームデータ処理装置。
【請求項９】
請求項８記載のストリームデータ処理装置であって、
前記第一及び第二の値式の集合における要素として共に任意の値式 Ｆがある場合に、前記値式 Ｆを第三の値式の集合における要素とする、
ことを特徴とするストリームデータ処理装置。
【請求項１０】
請求項８記載のストリームデータ処理装置であって、
第一の値式の集合の要素である任意の値式 Ｆ、第二の値式の集合の要素である任意の値式 Ｇに対して、Ｆの値が同じストリームデータは必ずＧの値が同一になることが示されている場合に、第三の値式の集合における要素をＧとする、
ことを特徴とするストリームデータ処理装置。
【請求項１１】
請求項８記載のストリームデータ処理装置であって、
前記第三の値式の集合と、前記第二のノードの数、前記第三のノードの数、ストリームデータに関する情報から、第三のノードの集合にデータを振分ける方法を示す前記第一のテーブルを生成し、
前記第三のノードの集合における振分け方法と前記第一の値式の集合、前記第二のノードの数、ストリームデータに関する情報から、第二のノードの集合のデータの振り分け方を、第三のノードの集合の振分け方と同様の方法または、第三のノードの集合より多くのノードにデータを振り分ける方法とし、
第二のノードの集合のデータの振り分け方法を示す前記第二のテーブルを生成する、
ことを特徴とするストリームデータ処理装置。
【請求項１２】
請求項８記載のストリームデータ処理装置であって、
前記第一のノードまたはクライアントのノードは、前記第一のテーブルから前記第三のノ

ードの集合を検索し、前記第三のノードの集合にデータと前記クライアントに関する情報を送り、
前記第三のノードの集合は、取得した前記クライアントの情報に基づいてデータを前記クライアントに転送する、
ことを特徴とするストリームデータ処理装置。
【請求項１３】
請求項１１に記載のストリームデータ処理装置であって、
前記第一のノードは、前記クライアントから、出力するデータ間で許容する入力された順序の時刻ずれ幅が示された場合に、異なる前記第三のノードの集合から出力されるデータに対して、前記第二のノードの集合または前記第三のノードの集合は、前記第三のノードの集合から出力されるデータのタイムスタンプ値及び許容する時刻ずれ幅を用いて出力時刻を調整し、入力されたデータの順序のずれを示された範囲に抑制する
ことを特徴とするストリームデータ処理装置。
【請求項１４】
請求項８記載のストリームデータ処理装置であって、
前記第一ノードは、複数のタイムスタンプ付加ノードを有し、同じ前記第三ノードで処理をするデータは必ず同じ前記タイムスタンプ付加ノードでタイムスタンプ付加処理をするように、複数の前記タイムスタンプ付加ノードにデータを振分ける方法を示すテーブルを生成する、
ことを特徴とするストリームデータ処理装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ストリームデータ処理システムにおいて、データごとに複数ノードでオペレータ処理や出力処理をする技術に関する。
【背景技術】
【０００２】
時々刻々と到着する大量のデータをリアルタイム処理するデータ処理システムに対する要求が高まっている。例えば、株自動売買，カープローブ，Webのアクセス監視，製造監視などを挙げることができる。
【０００３】
従来、企業情報システムのデータ管理の中心にはデータベース管理システム（Data Base Management System、以下、ＤＢＭＳとする）が位置づけられていた。ＤＢＭＳは、処理対象のデータをストレージに格納し、格納したデータに対してトランザクション処理に代表される高信頼な処理を実現している。しかし、ＤＢＭＳでは、新たなデータが到着する度に、全てのデータに対して検索処理を施すため上記のリアルタイム性の要求を満たすことは難しい。例えば、株取引を支援する金融アプリケーションを考えた場合、株価の変動にいかに迅速に反応できるかがシステムの最重要課題の一つである。しかし、データの検索処理が株価変動のスピードに追いつくことができず、ビジネスチャンスを逃してしまうことになりかねない。
【０００４】
このようなリアルタイムデータ処理に好適なデータ処理システムとして、ストリームデータ処理システムが提案されている。例えば非特許文献１にストリームデータ処理システム“ＳＴＲＥＡＭ”が開示されている。ストリームデータ処理システムでは従来のＤＢＭＳとは異なり、まずクエリ（問合せ）をシステムに登録しデータの到来と共に該クエリが継続的に実行される。実行されるクエリがあらかじめ把握できるため、新たなデータが到着したら、それまでの処理結果からの差分のみを処理することで高速な処理が可能である。したがって、ストリームデータ処理によって、株取引などにおける高レートで発生するデータをリアルタイムに解析し、ビジネスに有効なイベントの発生を監視して利活用することが可能になる。

【０００５】
大量のデータに対して高速処理するために、ストリームデータ処理では複数のコンピュータ（ノード）による分散処理が求められている。分散処理では、クエリを構成するオペレータごとに異なるノードで処理する方式（以下、パイプライン並列方式とする）と、同一のオペレータに対してデータごとに複数ノードで処理する方式（以下、データ並列方式とする）が知られている。特に、データ並列方式では、パイプライン並列方式と比してノード数の増加に従って通信オーバーヘッドはさほど増加しないため、スループットを大幅に向上させることが可能である。
【０００６】
データ並列方式においてデータの各ノードへの割り当て方法は、各オペレータの処理方法から算出される。ストリームデータ処理記述用の言語は非特許文献２に開示されているＣＱＬ（Ｃｏｎｔｉｎｕｏｕｓ　Ｑｕｅｒｙ　Ｌａｎｇｕａｇｅ）等、ＤＢＭＳで広く用いられているＳＱＬ（Ｓｔｒｕｃｔｕｒｅｄ　Ｑｕｅｒｙ　Ｌａｎｇｕａｇｅ）に近い言語で書かれていることが多く、ＲＤＢに準じた方法でデータの分割方法を計算することが可能である。例えば、ＣＱＬにはＳＱＬと同様にJoin（結合）オペレータやAggregation（集計）オペレータが存在し、ＲＤＢと同様に、それぞれ結合の条件や集計の単位によってデータの分割の仕方が定まる。ストリームデータ処理システムにおけるデータ並列化方式としては、特許文献１、特許文献２、非特許文献３、非特許文献４、非特許文献５が開示されている。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】米国公開特許US2007/0288635号
【特許文献２】米国公開特許US2008/0168179号
【非特許文献】
【０００８】
【非特許文献１】Ｒ．　Ｍｏｔｗａｎｉ，Ｊ．　Ｗｉｄｏｍ，Ａ．　Ａｒａｓｕ，Ｂ．　Ｂａｂｃｏｃｋ，Ｓ．　Ｂａｂｕ，Ｍ．　Ｄａｔａｒ，Ｇ．　Ｍａｎｋｕ，Ｃ．　Ｏｌｓｔｏｎ，Ｊ．　Ｒｏｓｅｎｓｔｅｉｎ　ａｎｄ　Ｒ．　Ｖａｒｍａ著：“Ｑｕｅｒｙ　Ｐｒｏｃｅｓｓｉｎｇ、Ｒｅｓｏｕｒｃｅ　Ｍａｎａｇｅｍｅｎｔ、ａｎｄ　Ａｐｐｒｏｘｉｍａｔｉｏｎ　ｉｎ　ａ　Ｄａｔａ　Ｓｔｒｅａｍ　Ｍａｎａｇｅｍｅｎｔ　Ｓｙｓｔｅｍ”，Ｉｎ　Ｐｒｏｃ．　ｏｆ　ｔｈｅ　２００３　Ｃｏｎｆ．　ｏｎ　Ｉｎｎｏｖａｔｉｖｅ　Ｄａｔａ　Ｓｙｓｔｅｍｓ　Ｒｅｓｅａｒｃｈ　（ＣＩＤＲ），Ｊａｎｕａｒｙ　２００３
【非特許文献２】Ａ．　Ａｒａｓｕ，Ｓ．　Ｂａｂｕ　ａｎｄ　Ｊ．　Ｗｉｄｏｍ著：“Ｔｈｅ　ＣＱＬ　ｃｏｎｔｉｎｕｏｕｓ　ｑｕｅｒｙ　ｌａｎｇｕａｇｅ：　ｓｅｍａｎｔｉｃ　ｆｏｕｎｄａｔｉｏｎｓ　ａｎｄ　ｑｕｅｒｙ　ｅｘｅｃｕｔｉｏｎ”，Ｔｈｅ　ＶＬＤＢ　Ｊｏｕｒｎａｌ，Ｖｏｌｕｍｅ　１５，Ｉｓｓｕｅ　２，ｐｐ．　１２１－１４２　（Ｊｕｎｅ　２００６）
【非特許文献３】T. Johnson, M. S. Muthukrishnan, V. Shkapenyuk, O. Spatscheck著：“Query-aware partitioning for monitoring massive network data streams”, SIGMOD, 2008
【非特許文献４】M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, M. J. Franklin著：“Flux: an adaptive partitioning operator for continuous query systems”, ICDE, 2003
【非特許文献５】M. Ivanova, T. Risch著：“Customizable parallel execution of scientific stream queries”, VLDB, 2005
【発明の概要】
【発明が解決しようとする課題】
【０００９】

前述のように、ストリームデータ処理ではまずクエリを登録しデータの到来と共にクエリが実行される。そして、システムに到着したデータは、途中の処理に影響されることなくその到着順を守って結果が出力される。したがって、データ並列化方式では到着順を守るため、システムに到着したデータはまず、タイムスタンプが付加され、その後、複数ノードでオペレータ処理される。そして、データ出力時にタイムスタンプ値に従った並び替えが行われる。

【００１０】

出力時の並び替えは、全てのデータに対して一つのノードで処理する。したがって、オペレータ処理のためのノード数を増加させても、出力時の処理によって律速されスループットを一定以上向上させることができない。また、一つのコンピュータの処理遅延によって他のコンピュータで処理した結果が遅延し、レイテンシが増大する。

【００１１】

実際のアプリケーションでは、一部のデータの間で、入力された順序を保証する必要がある場合が多い。そのようなアプリケーションに対して必要部分で入力された順序を保ちつつ高速に実行するために、複数のクエリを定義し、手動で必要に応じて各クエリの出力データをマージする。しかし、この方式では、ユーザが各クエリの出力を把握していなければ、性能だけでなくデータ順序の整合性を保つこともできず利便性が低い。

【００１２】

本願発明の目的は、順序を保証すべきデータに対して順序を保ちつつ高速にストリームデータ処理を行うストリームデータ処理方法及び装置を提供することにある。

【課題を解決するための手段】

【００１３】

上記の目的を達成するため、本発明においては、複数のノードにより、入力されるデータに対する処理を行うストリームデータ処理方法及び装置であって、第一のノードでクエリの登録を行い、第二のノードの集合でタイムスタンプが付加されたデータをオペレータ処理し、第三のノードの集合でオペレータ処理されたデータを前記タイムスタンプに従って並び替えて出力し、第一のノードは、登録したクエリから、入力された順序を同じグループのデータで保証するというグループが示された第一の値式の集合を抽出し、オペレータ処理を複数の第二のノードでデータごとに分割する方法を示す第二の値式の集合を算出し、且つ複数の第三ノードにデータを振分ける方法を解析し、同じ第二のノードで処理されたデータが同じ第三のノードで処理されるようにデータを振り分けるストリームデータ処理方法及び装置を提供する。

【００１４】

すなわち、本発明では，クエリの定義において、ユーザはグループ（以下、順序保証グループと呼ぶ）を指定する。そして、同じ順序保証グループのデータ間においては入力された順序が保証される。同じ順序保証グループに属するデータを同一のノードで出力することで、必要部分に対して入力された順序を保証する。また、異なる順序保証グループに属するデータを別のノードで出力することで、複数ノードでの処理を実現し性能低下を回避する。また、ユーザが指定した順序保証グループから、自動的にクエリの出力処理方法が解析されるので実現が容易である。

【００１５】

本発明の好適な実施態様では、ストリームデータ処理サーバにクエリの出力処理を並列化する機構（以下、クエリ並列化部と呼ぶ）を追加する。そして、順序保証グループが指定されたクエリがサーバに登録されると、クエリ並列化部が動作する。まず、クエリのオペレータをデータ並列方式で処理する方法を算出した後に、順序保証グループを読み込み、算出したオペレータの処理方法と組み合わせ計算する。そして、クエリの出力処理における複数ノードへのデータの振分け方法を解析する。また、出力処理の振分け方から、オペレータ処理における複数ノードへのデータの振分け方法を解析する。

【００１６】

また、出力ノードを管理する機構（以下、出力ノード管理部）をストリームデータ処理

エンジンに追加する。そして、クエリ実行時に、オペレータ処理の振分け方法にしたがって複数ノードでオペレータを処理すると共に、出力ノード管理部でクライアントからデータ出力要求を受け取る。そして、出力処理の振分け方法から、データを出力するノードを検索し、クライアントと出力データの情報を出力ノードに伝える。最後に、出力ノードでは対応するデータのオペレータ処理が完了するとクライアントに転送する。

【発明の効果】

【００１７】

本発明により、入力された順序を必要に応じて保証しつつ、高スループットかつ低レイテンシなストリームデータ処理を実現できる。

【図面の簡単な説明】

【００１８】

【図１】各実施例に利用される計算機システムの一例を示すブロック図である。

【図２】第１実施例の計算機システムの動作図である。

【図３】第１実施例における、クエリ定義の一例を示す図である。

【図４】第１実施例におけるクエリ登録時動作のフローチャート図である。

【図５】第１実施例における、オペレータ処理分割キー算出のフローチャートを示す図である。

【図６】第１実施例における、出力処理分割キー算出のフローチャートを示す図である。

【図７】第１実施例における実行木、実行ノード参照テーブル、出力ノード参照テーブル生成のフローチャート図である。

【図８】第１実施例における、クエリ登録時処理の動作例を示す図である。

【図９】第１実施例における、実行木、実行ノード参照テーブル、出力ノード参照テーブル配布処理の動作例を示す図である。

【図１０】第１実施例における出力処理分割キー算出の動作例を示す図である。

【図１１】第１実施例におけるクエリ実行時のフローチャートを示す図である。

【図１２】第１実施例における、クエリ実行時の動作例を示す図である。

【図１３】第１実施例における、クライアントの出力ノード検索処理のフローチャートを示す図である。

【図１４】第１実施例における、クライアントの出力ノード検索処理の動作例を示す図である。

【図１５】第１実施例における、実行時出力ノード変更処理を示す図である。

【図１６】第１実施例における、車渋滞検出クエリ処理のクエリ定義の一例を示す図である。

【図１７】第１実施例における、車渋滞検出クエリ処理のクエリ登録時の動作例を示す図である。

【図１８】第１実施例における、車渋滞検出クエリ処理のクエリ実行時の動作例を示す図である。

【図１９】第２実施例における、出力ノード制御方式の動作図である。

【図２０】第２実施例における、実行ノード制御方式の動作図である。

【図２１】第２実施例における、クエリ定義の一例を示す図である。

【図２２】第２実施例における、出力ノード制御方式のフローチャートを示す図である。

【図２３】第２実施例における、出力ノード制御方式の動作例を示す図である。

【図２４】第２実施例における、実行ノード制御方式のフローチャートを示す図である。

【図２５】第２実施例における、実行ノード制御方式の動作例を示す図ある。

【図２６】第３実施例における、計算機システムの動作図である。

【図２７】第３実施例における、フローチャートを示す図である。

【図２８】第３実施例における、クエリ登録時の動作例を示す図である。

【図２９】第３実施例における、クエリ実行時の動作例を示す図である。

【図３０】第３実施例における、クライアントによる振分け処理のフローチャートを示す図である。

【図３１】第３実施例における、クライアントによる振分け処理の動作例を示す図である。

【発明を実施するための形態】

【００１９】

以下、図面に基づき本発明の種々の実施形態を説明する。

【００２０】

図１は各実施例が適用される計算機システムの一構成を示す図である。メモリ１５０上に入力データ送信部２６０を含む入力クライアントノード２０１、メモリ１５１上に出力データ受信部２６１を含む出力クライアント２０４は、ネットワーク１２７を介してストリームデータ処理システムが稼動するストリームデータ処理サーバ２０６に接続されている。ネットワーク１２７は、イーサネット（登録商標）、光ファイバなどで接続されるローカルエリアネットワーク（ＬＡＮ）、もしくはＬＡＮよりも低速なインターネットを含んだワイドエリアネットワーク（ＷＡＮ）でも差し支えない。また、クライアント計算機２０１、２０４は、パーソナルコンピュータ（ＰＣ）、ブレード型の計算機システムなどの任意のコンピュータシステムでよい。

【００２１】

ストリームデータ処理サーバ２０６は、メインノード２０７、複数の実行ノード２３４、複数の出力ノード２４６で構成され、各ノードは外部ネットワーク１２７、もしくは、内部ネットワークを介して接続されている。ノード２０７、２３４、２４６は、インタフェース部を構成するＩ／Ｏインタフェース１３６～１３８、それぞれ処理部を構成する中央処理部（ＣＰＵ）１３０～１３２、記憶部であるメモリ１３９～１４１が、バスで結合された計算機であり、ブレード型計算機システム、ＰＣサーバなどの任意のコンピュータシステム、データ処理装置でよい。

【００２２】

ノード２０７、２３４、２４６はそれぞれＩ／Ｏインタフェース１３６～１３８を介してクライアント計算機２０１、２０４にアクセスする。ノード２０７、２３４、２４６で、ストリームデータ処理結果、処理の中間結果、システム動作に必要な設定データを不揮発性のストレージに格納する場合には、それぞれ記憶部であるストレージ装置１３３～１３５を用いることができる。ストレージ装置１３３～１３５は、それぞれＩ／Ｏインタフェース１３６～１３８を介して直接接続される、もしくはそれぞれＩ／Ｏインタフェース１３６～１３８よりネットワークを介して接続される。ノード２０７、２３４、２４６の各々のメモリ１３９～１４１に、ストリームデータ処理をする各ストリームデータ処理エンジン（ソフトウェア）１０３、１１５、１２０がマッピングされている。

【００２３】

なお、本明細書において、メインノード２０７、複数の実行ノード２３４、複数の出力ノード２４６をそれぞれ第一のノード、第二のノード、第三のノード、あるいは第一のノードの集合、第二のノードの集合、第三のノードの集合と呼ぶ場合がある。また、以上の説明では、サーバ２０６に関して複数のノードで成り立つマルチコンピュータの構成を示したが、サーバの構成については一つのノードに複数のＣＰＵで構成されるマルチプロセッサコンピュータに複数のストリームデータ処理エンジン１０３、１１５、１２０が動作しても構わない。また、マルチコンピュータとマルチプロセッサコンピュータを組み合わせた構成でも構わない。

【００２４】

以下でストリームデータ処理エンジン１０３、１１５、１２０の各モジュールを説明する。ストリームデータ処理エンジン１０３には、クエリ並列化部２０９を含むクエリ登録部２０８、出力ノード参照テーブル格納領域１０９、実行ノード参照テーブル格納領域１１０、データ入力部２１２、出力ノード管理部２１３がある。通常、ストリーム処理サーバ２０６の第一のノード２０７のストリームデータ処理エンジン１０３において、入力したデータに対してタイムスタンプ付加の処理を行うが、ここでは説明の煩雑さのため説明を省略する。一方、ストリームデータ処理エンジン１１５には、クエリ実行部２４０、実

行木格納領域１１７がある。ストリームデータ処理エンジン１２０には、データ出力部２４８がある。なお、各モジュールの動作については後述する。

【実施例１】

【００２５】

以下、第１実施例について図面に基づき説明する。

【００２６】

図２は第１実施例の動作を説明するための図である。アプリケーションによっては、クエリが処理するデータに対し必ずしも入力された順序を保証することが求められない。第１実施例では、そのような場合に、あらかじめ指定させたグループが同じであるデータのみ、入力された順序を保証することでスループットの向上及びレイテンシの削減を実現する。

【００２７】

まず、ユーザによって定義されたクエリ２０３はメインノード２０７が受信する。ユーザ定義クエリ２０３はデータの処理方法の記述に加え、処理するデータが属する順序保証グループが指定される。同じ順序保証グループに属するデータは入力された順序が保証される。メインノード２０７ではクエリ登録部２０８がクエリの各オペレータを解釈すると共に、クエリ登録部２０８のクエリ並列化部２０９は、順序保証グループ及び、クエリのデータ並列実行方法を用いて、オペレータ処理及び出力処理を複数のノードで行うためのデータの振分け方法を解析する。そして、得られたデータ振分け方法から、実行木２３７～２３９、実行ノード参照テーブル２６０、及び出力ノード参照テーブル２１１を生成する。実行木２３７～２３９はそれぞれ実行ノード２３４～２３６において処理するオペレータを記述したもので、出力ノード参照テーブル２１１はデータのレコード名をインデックスとして、各データを出力するノード名を参照するテーブルである。実行ノード参照テーブル２６０はデータのレコード名をインデックスとして、各データをオペレータ処理するノード名を参照するテーブルである。クエリ登録部２０８は実行木２３７～２３９をそれぞれ実行ノード２３４～２３６の実行木格納領域に格納し、出力ノード参照テーブル２１１をメインノード２０７の出力ノード参照テーブル格納領域に格納する。また、実行ノード参照テーブル２６０をメインノード２０７の実行ノード参照テーブル格納領域に格納する。

【００２８】

そして、入力クライアント２０１の入力データ送信部２６０からメインノード２０７のデータ入力部２１２へ入力データ２０２を転送することで登録したクエリの実行を開始する。データ入力部２１２は入力データ２０２にタイムスタンプを付加し、データを実行ノード参照テーブル２６０に従って実行ノード２３４～２３６に振り分ける。実行ノード２３４～２３６はそれぞれクエリ実行部２４０～２４２がデータを受信し、実行木２３７～２３９で指定されたオペレータで処理する。そして、オペレータ処理により実行ノード２３４、２３５でそれぞれ生成された計算データ２４３、２４４を出力ノード２４６に転送し、実行ノード２３６で生成された計算データ２４５を出力ノード２４７に転送する。

【００２９】

なお、実行ノード２３４～２３６で生成された計算データ２４３～２４５は、実行木によっては出力ノードに直接転送されない。実行木２３７によっては、計算データ２４３を実行ノード２３５、２３６の実行ノードに転送し、クエリ実行部２４１、２４２でオペレータ処理する。そして、オペレータ処理により生成された計算データを出力ノード２４６、２４７に転送する。計算データ２４４、２４５についても同様に実行ノード２３４～２３６に転送される。

そして、最終的に出力ノード２４６、２４７が受信した計算データ２４３～２４４、２４５をそれぞれデータ出力部２４８、２４９で、計算データに付加されたタイムスタンプに従って並び替え、それぞれ出力データ２５２、２５３を生成する。

【００３０】

一方、出力データ２５２、２５３を取得する出力クライアント２０４は、取得する出力

データのレコード名（例えば、クエリ３０１ならば銘柄名としてＡ電、Ｂ銀、Ｈ電等）と出力クライアントの名前２０５をメインノード２０７に転送する。そして、メインノード２０７の出力ノード管理部２１３は、クエリ登録時に生成した出力ノード参照テーブル２１１によりレコード名に対応するデータを出力するノードの名前を検索する。そして、検出された出力ノード２４６、２４７にそれぞれ出力先のクライアントの名前と出力レコード名２５０、２５１を転送する。出力ノード２５０、２５１は、それぞれ出力データ２５２、２５３のレコード名と出力レコード名２５０、２５１とのマッチングをし、一致していたら出力クライアント２０４にそれぞれ出力データ２５２、２５３を転送する。

【００３１】

第１実施例では、既存のデータ並列方式で出力ノード処理がボトルネックになっている場合にスループットを向上させる。例えば、図１８に示すように、実行ノード数６、出力ノード数２で処理する時、一つの出力ノードにおける処理可能な入力レートをＴ（タプル／秒）とすると、既存のデータ並列方式では、全体で処理可能な入力レートもＴ（タプル／秒）になるのに対し、第１実施例では２Ｔ（タプル／秒）となる(一つの処理ノードの処理可能な入力レートがＴ／３(タプル／秒)以上の場合)。

【００３２】

続いて、第１実施例の動作の詳細を述べる。

【００３３】

ユーザによるクエリの定義方法について説明する。図３はユーザが定義したクエリの一例である。本実施例では、順序保証グループが示された値式の集合をクエリに対して指定させる。指定の方法は、クエリに記述しても、設定ファイルを介して指示されていてもその他の方法でも構わない。図３では、ストリームの定義において、ストリームデータ処理記述言語（ＣＱＬ）に新たに追加した構文 Order Partition Byを用いて値式 銘柄が指定されている。値式の指定は、ストリームの定義以外にもクエリ定義やその他の箇所で行われても構わない。

【００３４】

クエリ３０１では、銘柄、取引所、注文数、口座、業種をカラムとするストリーム 株注文を定義する。ストリーム 株注文は株取引に関する情報を配信しており、クエリq1はストリーム 株注文を入力し、銘柄及び取引所ごとの前一分間の合計注文数を常時出力する。

【００３５】

クエリ３０１を定義したユーザは、同じ銘柄に対しては取引の順序にしたがって集計結果を得る必要があるが、異なる銘柄に対して取引の順序を要求していないとする。この時、ユーザは、入力ストリーム 株注文の定義において、構文Order Partition Byを用いてストリームのカラム 銘柄を指定することで、入力された順序を異銘柄に対して保証する必要がないことをシステムに伝える。

【００３６】

ユーザによって定義されたクエリの登録時動作について述べる。図４はメインノードにおけるクエリ登録時のフローチャートである。最初に、ユーザ定義クエリを読み込み、全てのオペレータを複数のノードにデータを分けて並列処理する方法を示す値式の集合（以下、オペレータ処理分割キーと呼ぶ）を算出する。値式はストリームのカラムを項とする演算式であり、後述する値式も同様である（４００）。

【００３７】

そして、算出したオペレータ処理分割キーと、Order Partition By構文で指定された値式の集合（以下、OrderPartitionByキーと呼ぶ）から、オペレータ処理及び出力処理におけるデータの振分け方法を解析する。振分け方法は、１．出力処理においてデータを複数のノードに振り分ける。２．同じノードでオペレータ処理されたデータを必ず同じノードで出力処理する、を満たすことで、高スループットな出力処理を実現する。なお、アプリケーションが要求するスループットを満たすならば、同じノードでオペレータ処理されたデータを異なるノードで出力処理する場合があっても構わない。

【００３８】
動作としては、まず、オペレータ処理分割キーとOrderPartitionByキーから出力時の処理を複数のノードにデータを分けて並列に行う方法が示された値式の集合（以下、出力処理分割キーと呼ぶ）を算出する（４０１）。そして、算出したオペレータ処理及び、出力処理分割キーから、各々の処理における各ノードへのデータの振分け方を解析する。なお、データの振分け方法の解析は、数値解析による方法など、出力処理分割キーの算出をしない方法でも構わない。そして、その振分け方から実行木、実行ノード参照テーブル、出力ノード参照テーブルを生成し各ノードに配信する（４０２）。

【００３９】
図５は図４の４００の動作詳細を示すフローチャートである。まず、ユーザ定義クエリを読み込み（５０１）、クエリを構成する各オペレータが複数のノードにデータを分けて並列処理する方法を示す値式の集合（以下、処理分割キーと呼ぶ）を抽出する（５０２）。処理分割キーの抽出方法は既存のＲＤＢ(Relational Data Base)に従い、オペレータの処理方法から抽出する。例えば、オペレータ Joinなら結合の等価条件で指定された値式を処理分割キーの要素とし、オペレータ aggregationならGroup By構文で指定された値式を処理分割キーの要素とする。

【００４０】
そして、各オペレータの処理分割キーの積集合Opをオペレータ処理分割キーとする（５０３）。Opが空集合であれば、クエリのオペレータの処理を各ノードで並列に実行することが不可能であることを通知し登録動作を終了する。または、オペレータ処理において計算データをノード間で再割当てすることを許可する場合には、一部のオペレータの積集合をオペレータ処理分割キーとして抽出してもよい。なお、オペレータ処理分割キーは上記の方法で算出する以外にも、クエリを定義したユーザなどによって指定されたキーを抽出するなど、その他の方法で導出されても構わない。

【００４１】
図６は図４の４０１の動作詳細を示すフローチャートである。まず、Order Partition By構文で指定した値式の集合（OrderPartitionByキー） Orを抽出する（６０１）。そして、オペレータ処理分割キー Op、OrderPartitionByキー Orから出力処理分割キー Ouを算出する。まず、Op、Orに対し、F(c) ⊂ Op ∧ F(c) ⊂ Or（F(c)は、ストリームの任意のカラム cを項とする演算式）を満たすF(c) が存在する場合は（６０２）、F(c)をOuの要素にする。条件を満たすF(c) が複数存在する場合には全てのF(c)を要素とする（６０３）。また、c/n ⊂ Op ∧ c/m ⊂ Or（n, mは整数）を満たすcが存在する場合には（６０４）、c / s(n,m) （s(n,m)はnとmの最小公倍数）をOuの要素にする。条件を満たすcが複数存在する場合には全てのc / s(n,m)を要素とする（６０５）。

【００４２】
さらに、ユーザによってG(c2) including F(c1) （F(c1)、G(c2)はストリームの任意のカラム c1、c2を項とする値式）という関係が指定されている場合には、F(c1) ⊂ Op ∧ G(c2) ⊂ Orを満たすならば、 G(c2)をOuの要素にする。G(c2) including F(c1)とは、値式 F(c1)が同じストリームデータは必ず値式 G(c2)が同一になることを示す。条件が複数存在するなら、全てのG(c2)を要素とする（６０６、６０７）。そして、Ouが空集合でなければ（６０８）、Ouを出力処理の分割キーとして算出する（６１０）。Ouが空集合ならば出力処理のデータ並列化が不可能であることを通知し登録動作を終了する（６０９）。４０１の動作は、統計的解析など前述の方法以外でも、値式 F(c1)が同じストリームデータは必ず値式 G(c2)が同一になることを示されれば、G(c2)をOuの要素にして構わない（F(c1)、G(c2)はストリームの任意のカラム c1、c2を項とする値式で、F(c1) ⊂ Op ∧ G(c2) ⊂ Orを満足）。

【００４３】
図７は図４の４０２の動作詳細を示すフローチャートである。まず、ユーザから指定された実行ノード数 n、出力ノード数 m、値式F(c) （F(c)は、ストリームの任意のカラム cを項とする演算式）とそのとりうる値（または、値の数）を読み込む（７０１）。なお

、実行ノード数及び、出力ノード数はＯＳ等のシステム情報から取得するなど、その他の方法で取ってきても構わない。また、F(c)とそのとりうる値もデータから統計的に分析する等、その他の方法で取得しても構わない。そして、算出したオペレータ処理及び出力処理の分割キーを併せて用いることで、各ノードへの振り分け方法を決定し、実行木及び実行ノード参照テーブル、出力ノード参照テーブルを生成する。
まず、出力処理分割キーの要素である値式のとりうる値の数が、実行ノード数nよりも大きい場合には、データをn個の出力ノードに振り分ける方法を決定し、出力ノード参照テーブルを生成する。振り分け方法は、値式の大小や辞書順で一定の領域に区切って振り分ける方法や、mod（剰余演算）等のハッシュ関数で値式からハッシュ値を求め、ハッシュ値に従って振り分ける方法や、その他、データの偏りを考慮した振り分け方法など任意の方法で構わない。そして、オペレータ処理におけるデータの振り分け方も出力処理の振り分け方と同様の方法にし、実行ノード参照テーブルを生成する。また、同じ実行木をｎ個、生成する（７０４）。こうして、データをオペレータ処理した後、同じノードで続けて出力処理をするようにデータを振り分けることで、オペレータ処理及び出力処理の並列化によって高スループット化しつつ、オペレータ処理と出力処理の間におけるノード間通信を発生させず低レイテンシ化する。

【００４４】

また、出力処理分割キーの要素である値式のとりうる値の数が、実行ノード数nよりも小さい場合には、データを最大で m個の出力ノードに振り分ける方法を決定し、出力ノード参照テーブルを生成する（７０２）。振り分け方法は、値を一定の領域に区切って振り分ける方法や、ハッシュ関数でハッシュ値を求め、ハッシュ値に従って振り分ける方法や、その他、任意の方法で構わない。そして、出力処理の振分け方法では同じ出力ノードに振分けられるデータに対して、オペレータ処理では複数の実行ノードに振分けることで、オペレータ処理で最大 n個の実行ノードに振分けるよう方法を解析する。具体的には、オペレータ処理分割キーの要素である値式を用いて複数の領域に分ける、または、値式からハッシュ値を求め、ハッシュ値で分けるなどして、複数の実行ノードに振り分ける方法を解析する。そして解析結果から実行ノード参照テーブルを生成する。また、実行ノード数と同数の実行木を生成する（７０５）。一般的にオペレータ処理は出力処理よりも計算処理コストが大きいため、オペレータ処理を出力処理よりも処理するノードを増やすことでスループットが向上する。しかし、そのためにオペレータ処理と出力処理の間に通信が発生しレイテンシが増大する。スループットの向上よりもレイテンシを低くすることを優先するアプリケーションなどでは、オペレータ処理を出力処理と同様にデータを振り分けるなどその他の方法でも構わない。
最後に、生成された実行木と出力ノード参照テーブル、実行ノード参照テーブルを各ノードに転送する（７０６）。

【００４５】

図８、図９は、クエリ３０１を例にしたクエリ登録時の動作である。クエリ３０１はオペレータrange、group by、istreamから構成される（８０１）。オペレータrange、istreamについては全てのデータの処理を並列に実行可能であり、したがって、処理分割キーの要素として任意の値式を取る。また、オペレータgroup byは、集計の単位より、ストリーム 株取引のカラム 銘柄または取引所が異なっているデータに対しては並列に実行可能である。したがって、オペレータgroup byの処理分割キーは｛銘柄、取引所｝（８０２）となる（５０２の動作）。したがって、各オペレータの処理分割キーの積集合を取ることにより、オペレータ処理分割キーは{銘柄、取引所}（８０３）と算出される（５０３、５０３の動作）。

【００４６】

また、構文 Order Partition Byより、OrderPartitionBy キーは{銘柄} （８０４）と抽出される（６０１の動作）。したがって、オペレータ処理分割キーとOrderPartitionBy キーに共通の要素としてカラム 銘柄を含む値式 銘柄が存在するので、出力処理分割キーは ｛銘柄｝ （８０５）と算出される（６０２、６０３の動作）。

【００４７】
そして、算出した出力処理分割キー、ユーザから指定された実行ノード数 ２（８０６）、出力ノード数 ２（８１０）、出力処理分割キーの要素である「銘柄」のとりうる値の数 １０４３から出力処理の振分け方法を決定する（７０１の動作）。まず、実行ノード数よりも「銘柄」のとりうる値の数のほうが大きいので、２つの実行ノードにデータを振分ける。振分け方法としては、出力分割キーの要素である「銘柄」に対しハッシュ関数（文字列をビット列で表現した値に対し、２で除算した剰余を返す関数）で求めたハッシュ値を用いる。ハッシュ値が０であるデータをノード１に、ハッシュ値が１であるデータをノード２に振分ける。そして、その振り分け方法を出力ノード参照テーブル８０９として実現する（７０２の動作）。
さらに、出力処理を２個（実行ノード数）のノードに振分けることが可能であったので（７０３の動作）、オペレータ処理のデータの振分け方法を、出力処理のデータの振り分け方法と同様にし、実行ノード参照テーブルを生成する。また、2個（実行ノード数）の実行木（８０８）を生成する（７０５の動作）。

【００４８】
生成された出力ノード参照テーブル９０６は、ストリームデータ処理サーバ９０１のメインノード#0（９０２）の出力ノード参照テーブル格納領域９０５に格納され、実行ノード参照テーブル９２３は、実行ノード参照テーブル格納領域９２２に格納される。また、実行木９２０、９２１はそれぞれ実行・出力ノード＃１（９０７）、＃２（９１０）の実行木格納領域９１８、９１９に格納される（７０６の動作）。

【００４９】
クエリ３０１は１つのクエリで構成されていたが、実際のストリームデータ処理のクエリでは複数のクエリが連なっていることが一般的である。本実施例では、複数のクエリが連なっている場合にも同様の方法でオペレータ処理分割キーの算出が可能である。また、クエリ３０１では、オペレータ処理分割キーとOrderPartitionByキーの要素にストリームの共通のカラムを含む値式が存在した。しかし、たとえ共通のカラムを含む値式が存在しなかったとしても出力処理分割キーの算出が可能である。以下は、図１０のクエリ１００１を例にした出力処理分割キー算出の動作である。

【００５０】
図１０に示すように、クエリ１００１は、クエリq2、q3から成る。クエリq2はクエリq1と同じくストリーム 株注文を入力とし、取引所、銘柄ごとの分単位の合計注文数を出力する。そして、クエリq3はクエリq2で出力した銘柄毎の取引所ごとの合計注文数から銘柄毎の最大注文数を出力する。クエリq2ではオペレータ group byから処理分割キーが｛取引所、銘柄｝（１００２）と抽出され、クエリq3では同じくオペレータ group byにより処理分割キーが｛銘柄｝（１００３）と抽出される。したがって、クエリq2、q3の処理分割キーの積集合を取ることによりオペレータ処理分割キーは｛銘柄｝（１００４）と算出される（５０２、５０３の動作）。

【００５１】
また、クエリ１００１では、構文 Order Partition byによりストリームのカラム 業種を指定している。したがって、OrderPartitionByキーは｛業種｝（１００５）となり（６０１の動作）、オペレータ処理分割キー｛銘柄｝とは同じカラムを含む値式は存在しない。しかし、クエリ１００１ではさらに構文 includingで｛銘柄｝が指定されることで、｛銘柄｝が同じストリームデータは必ず｛業種｝が同じになることを示している。したがって、出力処理分割キーとして｛業種｝（１００６）が算出される（６０６の動作）。

【００５２】
クエリ実行時の動作詳細を示す。図１１は実行時動作のフローチャートである。
本実施例では、出力ノードが複数存在するため、出力クライアントが取得するデータが出力されるノードをあらかじめ把握できない。したがって、以下の動作をストリームデータ処理システムに追加する。

【００５３】

まず、出力クライアント２０４は、取得するデータのレコード名及びクライアント名をメインノード２０７に送信する（１１０１）。そして、メインノード２０７では、クエリ登録時に格納された出力ノード参照テーブル２１１によりレコード名から、対応するデータを出力するノードを検索する。（１１０２）。さらに、メインノード２０７では検出した出力ノードにレコード名と出力クライアント名を送信する（１１０３）。出力ノードでは、オペレータ処理された出力データのレコード名とメインノードから受信したレコード名を比較する（１１０４）。そして、レコード名が一致していたならば、メインノード２０７から転送された名前のクライアントにデータを出力する（１１０４）。

【００５４】

図１２は図３に示したクエリ３０１を例にしたクエリ実行時の動作である。クエリ３０１に対して入力クライアント２０１からデータが入力されると実行が開始される。まず、サーバ９０１のメインノード＃０（９０２）で入力データ１２０６－１２０９にタイムスタンプを付加し（１２１０）、付加されたデータを、実行ノード参照テーブル（出力ノード参照テーブル）９０６に従って処理・出力ノード ＃１（９０７）、＃２（９１０）に送信する。今、銘柄 Ａ電のハッシュ値を０、銘柄Ｈ電、Ｂ銀のハッシュ値を１とする。その時、銘柄 Ａ電のデータ１２２３は実行・出力ノード＃１（９０７）に送信し、銘柄Ｈ電、Ｂ銀のデータ１２１３は実行・出力ノード＃２（９１０）に送信する。そして、実行・出力ノード＃１、＃２では受信されたデータを各々実行木９０９で指定されたオペレータ（range、group by、istream）で処理する。

【００５５】

一方、出力クライアント１．２．３．４（ＩＰアドレス）（１２０５）では取得するデータのレコード名 Ａ電、Ｈ電、Ｂ銀及びクライアント名 １．２．３．４（１２１９）をメインノード＃０に送信する（１１０１の動作）。そして、メインノード＃０では出力ノード参照テーブル９０６を参照することで、受信したレコード名のデータを出力するノードを検索する（１２１１）。

【００５６】

すなわち、出力ノード参照テーブル９０６から銘柄名 Ａ電のデータは処理・出力ノード#1、銘柄名 Ｈ電、Ｂ銀のデータは実行・出力ノード＃２から出力されると検出される（１１０２の動作）。そして、各出力ノードに、出力クライアント名１．２．３．４と各々のノードに対応する銘柄名を送信する。実行・出力ノード#1は「銘柄 Ａ電」（１２２０）が送信され、実行・出力ノード＃２には「銘柄 Ｂ銀、Ｈ電」（１２２２）が送信される（１１０３の動作）。

【００５７】

実行・出力ノード＃１ではオペレータを処理（１２１７）後に、メインノード＃０からの情報１２２０に基づいて、銘柄名 Ａ電のデータ（１２２１）が出力クライアント１．２．３．４に出力される。また、＃２では、オペレータを実行（１２１８）後、＃０からの情報１２２２に基づいて、銘柄名 Ｂ銀、Ｈ電のデータ（１２２３～１２２５）が出力クライアント１．２．３．４に出力される（１１０４の動作）。

【００５８】

続いて、第１実施例における拡張方式を述べる。まず、クライアント側の出力ノード管理について説明する。基本動作では出力ノード参照テーブル２１１をメインノード２０７に格納し、その出力ノード参照テーブル２１１を用いて出力ノードを検索した。しかし、出力ノード導出処理はサーバのノードではなく出力クライアントで行っても構わない。ストリームデータ処理においては出力先のクライアントが多数存在することも多く、そのような場合にはサーバで出力ノードを検索するのと比べて、新たにサーバ・ノードを追加することなしに出力ノード管理を並列実行させることができる。図１３はその拡張方式のフローチャートである。

【００５９】

まず、図１３の上段において、クエリの実行開始時に出力クライアント２０４はメインノード２０７に出力ストリームの名前を送信する（１３０１）。そして、メインノード２

０７では、受信した出力ストリーム名に対応する出力ノード参照テーブルを出力クライアントに送信する（１３０２）。最後に、出力クライアントでは受信した出力ノード参照テーブルを出力データ受信部に格納する（１３０３）。

【００６０】

次に、図１３下段に示すクエリ実行時に、出力クライアントは出力ノード参照テーブルを参照し、取得をするデータのレコード名から出力ノードを検索する（１３０４）。そして、対応する出力ノードにデータのレコード名とクライアント名を送信する（１３０５）。出力ノードでは、オペレータ実行されたデータのレコード名と、出力クライアントから受信したレコード名を比較し一致すると、そのデータを出力クライアントに出力する（１３０６）。

【００６１】

図１４は、クエリ３０１の例を用いたクライアントによる出力ノード管理動作である。

【００６２】

まず、図１４上段に示すクエリの実行開始時に、出力クライアント １．２．３．４（１２０５）は出力ストリーム名 ｑ1（１４０２）をメインノード ＃０（９０２）に転送する（１３０１の動作）。#0ではｑ1の登録時に、出力ストリーム名 ｑ1（１４０２）に対応する出力ノード参照テーブルを生成し出力ノード参照テーブル格納領域に格納する。その格納されているテーブルを１．２．３．４に転送する（１３０２の動作）。

【００６３】

そして、図１４下段に示すクエリ実行時に、１．２．３．４は出力ノード参照テーブル９０６を参照し、取得する銘柄 Ａ電のデータの出力ノードを＃１、同じく取得する銘柄 Ｈ電、Ｂ銀のデータの出力ノードを＃２と検出する（１３０４の動作）。そして、１．２．３．４はクライアント名「１．２．３．４」の情報（１４１１、１４１２）と共に、＃１に「銘柄 Ａ電」の情報（１４１１）、そして、＃２に「銘柄 Ｈ電、Ｂ銀」の情報（１４１２）を送信する（１３０５の動作）。＃１、＃２では、入力データに対してオペレータを処理後、＃１は「銘柄 Ａ電」の情報（１４１１）より銘柄名 Ａ電のデータ（１４０９）を1.2.3.4に出力し、＃２は「銘柄 Ｈ電、Ｂ銀」の情報（１４１２）より銘柄名 Ｈ電、Ｂ銀のデータ（１４１０）を１．２．３．４に出力する（１３０６の動作）。

【００６４】

図１５は、実行時に出力ノードを変更する処理に関する拡張方式である。ストリームデータ処理システムでは、実行時にデータが入力される。そして、データの入力レートが大きくなる、または、入力されるデータの種類に偏りが生じる等でノードが過負荷になるため、実行時に出力ノードにおけるデータの振分けの仕方を変更する必要が生じる。

【００６５】

クライアントに出力ノード参照テーブルを配置する場合の実行時出力ノード変更方法を述べる。図１５上段のフローにおいて、データの割振り方法を変更する時に、まず、メインノードにある出力ノード参照テーブルが更新され、その更新された出力ノード参照テーブルとその出力ストリームの名前を出力クライアントに送信する（１５０１）。そして、出力クライアントでは、受信した出力ストリーム名に対応する出力ノード参照テーブルを受信したテーブルに置換える（１５０２）。

【００６６】

図１５下段により、クエリ３０１を例にした実行時出力ノード変更処理の動作を述べる。変更前、＃１に銘柄%２が０であるデータ、＃２に銘柄％２が１であるデータが振分けられている。そして、変更によって新たに出力ノード＃３が追加され、＃２に銘柄％４が１であるデータ、＃３に銘柄％４が３であるデータが割当てられたとする。その時、メインノード＃０（９０２）で更新された出力ノード参照テーブル１５０６を出力ストリーム名q1（１４０２）と共に出力クライアント１．２．３．４（１２０５）に転送する（１５０１の動作）。そして、１．２．３．４では出力ストリーム名 q1の出力ノード参照テーブル９０６を更新された出力ノード参照テーブル１５０６に置き換える（１５０２の動作）。その後は、１．２．３．４では置換された出力ノード参照テーブルを参照することで

、取得データの出力ノードを把握する。

【００６７】

続いて、車渋滞検出クエリの例を用いて第１実施例の動作とその効果を述べる。特に、最小公倍数を用いて出力処理分割キーを算出する動作（６０５、６０６の動作）を説明する。

【００６８】

図１６は車渋滞検出クエリである。クエリでは、カラムとして車両ＩＤ、ｘ、ｙを持つストリーム 車両情報１６０１を入力とする。車両ＩＤは車両ごとに割当てられており、ｘ、ｙはその各車両の位置情報を表しており、ストリーム 車両情報では定期的に各車両の位置情報が配信される。クエリでは各々の車両の現在と一つ前の位置情報（ｘ、ｙ）から車両速度を計算する（１６０４）。また、複数の車両の位置情報により各々の車両の周りの車両密度を計算する（１６０５）。そして、車両速度と車両密度から各々の車両周辺の渋滞量を計算する（１６０６）。

【００６９】

上記のクエリに対し、ユーザは、近くを走行中の車両同士の渋滞情報は時刻順に更新されることを求めているとする。すなわち、入力ストリーム 車両情報（１６０１）に対して、構文 Order Partition Byで(int)x/300を指定し、(int)x/300ごと、すなわち、xがそれぞれ０～２９９、３００～５９９の範囲でそれぞれ渋滞情報を順序通りに処理することをシステムに伝える。

【００７０】

図１７はクエリ１６０２を用いた登録時の動作例である。

【００７１】

まず、オペレータ処理分割キーを算出する。クエリ１６０２は、車両速度計算（１６０４）、車両密度計算（１６０５）、渋滞量計算（１６０６）に関するクエリで構成され、それぞれのクエリの処理分割キーは、クエリの各オペレータの処理分割キーの積集合を取ることで、{車両ID, (int)x/100}（１７０１）、{(int)x/100}（１７０２）、｛（任意）｝（１７０３）となる（５０２、５０３の動作）。したがって、オペレータ処理分割キーは各クエリの処理分割キーの積集合{(int)x/100}（１７０４）となる（５０３、５０４の動作）。

次に、出力処理分割キーを算出する。まず、OrderPartitionByキーが、Order Partition By構文より{(int)x/300}（１７０５）と求まる（６０１の動作）。そして、オペレータ処理分割キー{(int)x/100}とOrderPartitionByキー{(int)x/300}は、ともに入力ストリームのカラムxを含む値式(int)x/100、(int)x/300を要素に持ち、かつ、共にx/n（nは整数）の形式を取るので、100と300の最小公倍数をとることにより出力処理の分割キーは{(int)x/300}（１７０６）と算出される（６０４、６０５の動作）。

【００７２】

そして、各処理キーからデータの割り当て方法を解析する。まず、ユーザによって指定された実行ノード数 ６（１７１１）、出力処理分割キーの要素の値式である(int)x/300の取りうる値数 ２（１７０８）という情報より（７０１の動作）、実行ノード数の方が値式の取りうる値の数よりも大きいので、最大で出力ノード数 ２個（１７０７）のノードに振り分ける方法を決定し、出力ノード参照テーブル１７１０を生成する（７０２の動作）。そして、オペレータ処理において最大６個（実行ノード数）に振分ける方法を解析する。今、オペレータ処理分割キー{(int)x/100}では最大６ノードにデータを割当てることが可能のため、オペレータ処理を６ノードにデータを振分ける方法を決定し、実行ノード参照テーブル１７１２を生成する。また、実行木１７０９を６個（実行ノード数分）生成する（７０５の動作）。最後に、生成した実行木１７０９を実行ノード＃１～＃６に配布し、出力ノード参照テーブル１７１０及び実行ノード参照テーブル１７１２をメインノード＃０に配布する（７０６の動作）。

【００７３】

図１８はクエリ１６０２を用いた実行時の動作例である。前述したようにクエリ１６０

２では、＃１～６（１８０２－１８０７）をオペレータ処理するノードとし、＃７（１８０８）、＃８（１８０９）を、オペレータ処理結果を出力するノードとする。メインノード #0で、実行ノード参照テーブル１７０９を用いて振り分けられたデータは、＃１～６で車両速度計算、車両密度計算（１８１０）、渋滞量計算（１８１１）に関するクエリによって実行される。例えば、＃３ではxが２００～３００の範囲の車両情報データが入力され、１８１２のように渋滞情報データが出力される。また、＃４ではｘが３００～４００の範囲のデータが処理され１８１５のように出力される。そして、＃１～３で出力された渋滞情報データは＃７に送信され、＃４～６の出力データは＃８に送信される。#7、#8ではその送信データを入力順に並び替え各々クライアントに出力する。例えば、＃７では＃１～＃３の最新データのタイムスタンプ値を比較し（１８１６）、最もタイムスタンプが古いデータ（１８１７）をクライアントに出力する（１８１８）。また、＃８では＃４～＃６の最新データを比較することで（１８１９）、同じく最もタイムスタンプ値が古いデータ（１８２０）をクライアントに出力する（１８２１）。

【実施例２】

【００７４】

次に第２の実施例について述べる。第１実施例では、入力された順序を保証するのを指定した順序保証グループ内に限定することで高スループット、低レイテンシを実現した。したがって、ノード間の負荷の違い等によって異なるデータグループでは時刻が大幅にずれることがある。しかし、アプリケーションによっては入力された順序を厳密に守る必要はないものの、大きくずれることは好まれないことがある。例えば、株取引の集計に関するクエリでは、複数のユーザが複数の銘柄情報を参考に投資行動を決めることを想定すると，銘柄別に出力時刻が大幅にずれることは，正確性，公平性の観点から回避が必要である。Ａ電株がＢ銀株よりも情報が遅く入った場合には、Ａ電株を見ている投資家はＢ銀株を見ている投資家より行動が遅れる。また、Ａ電株とＢ銀株を比べて有利な行動を選択しようとする投資家は判断を誤る。また、車渋滞検出のクエリにおいては、指定した順序保証グループの境界（クエリ１６０２ならｘ＝３００）で順序が保たれない。したがって、例えば、データグループの境界付近を走行中の車両は渋滞情報が時系列順に表れないことがあるため、カーナビで表示する場合に見にくくなるだけでなく、走行経路の選択を誤ることにもつながる。

【００７５】

そこで、第２実施例では、順序保証グループが異なるデータ間に対しても時刻のずれ幅の一定以内に抑制する。図１９、図２０はそれぞれ第２実施例の構成を示す。図１９は出力ノードでずれ幅を制御する方式を表しており、図２０は処理ノードでずれ幅を制御する方式を表している。出力ノード制御方式ではデータ出力時、処理ノード２４６、２４７におけるストリームデータ処理エンジンのデータ処理部２４８、２４９に出力時刻調整部１９０１、１９０２を追加し、お互いの出力データの時刻を交換することで出力時刻を調整する。一方、実行ノード制御方式では、実行ノード２３５のストリームデータ処理エンジンに追加した出力時刻調整部２００１で調整する。つまり、クエリ実行部２４１で処理された計算データ２４４を出力ノード２４６と出力２４７の両方に、送信時刻を調整しつつ転送する。

【００７６】

第２実施例では、時刻のずれ幅を一定以内に抑制しつつ、既存のデータ並列方式よりもスループットを向上させる。例えば、実行ノード数６、出力ノード数２で処理する時(図２３、図２５)、一つの出力ノードにおける処理可能な入力レートをＴ（タプル／秒）とすると、既存のデータ並列方式では、全体で処理可能な入力レートがＴ（タプル／秒）になるのに対し、第２実施例では１．５Ｔ（タプル／秒）となる（一つの処理ノードの処理可能な入力レートがＴ／３(タプル／秒)以上の場合）。

【００７７】

以下、第２実施例の動作の詳細を述べる。第２実施例では、Order Partition By構文で指定したデータグループが異なるデータに対して許容する時刻ずれ幅を指定させるために

、ストリームデータ処理記述言語に新たに構文 limitを追加する。例えば、図２１におけるクエリ２１０１では、構文 Order Partition Byで｛銘柄｝をキーに指定しlimit構文で１秒と指定することで、異なる銘柄間においても出力時刻のずれが１秒以内となるようにする。許容する時刻ずれ幅の指定は、順序保証グループの指定と同様、ストリームの定義以外にもクエリ定義やその他の箇所で行われても構わない。Limit構文は、クエリ登録時にOrderPartitionBy構文と共に解釈され、実行木や出力ノード参照テーブルと同様に、許容時刻ずれ幅に関する情報を該当するノードに転送する。

【００７８】

図２２は出力ノード制御方式のフローチャートである。まず、実行ノードで第１実施例と同様にオペレータを処理する。そして、処理完了後（２２０１）、計算データを出力ノードに送信する（２２０２）。一方、出力ノードではお互いに出力データのタイムスタンプ値を交換する。そして、他の出力ノードからタイムスタンプ値を受信する（２２０３）、または、実行ノードから計算データを受信したら（２２０３）、各実行ノードから送信された最新データの時刻を比較し最古のデータを抽出する（２２０７）。そして、抽出データのタイムスタンプ値と，他ノードから転送された最新のタイムスタンプ値に許容時間ずれ幅を加算した時刻を比較する。そして、抽出データのタイムスタンプ値のほうが古ければ（２２０４）、クライアントにデータを出力する（２２０６）。またその際に、他の出力ノードに出力したデータのタイムスタンプ値を通知する（２２０５）。

【００７９】

図２３はクエリ２１０１を用いた出力ノード制御方式の動作例である。＃１～６（２３０２～２３０７）はオペレータを処理するノードであり、＃７,８（２３０８、２３０９）は処理結果を出力するノードである。＃１～３でオペレータ処理された計算データは＃７に転送され、＃４～６で処理されたデータは＃８に転送される。

【００８０】

今、＃３では銘柄 Ａ電のデータ（２３１０）が処理され（２２０１の動作）、その処理結果が＃７に出力される（２２０２の動作）。また、＃４では銘柄 Ｈ電、Ｂ銀のデータ（２３１１）が処理され（２２０１の動作）、その処理結果が＃８に出力される（２２０２の動作）。また、＃７ではデータ２３２１を出力し、データ２３２１のタイムスタンプ値は０’２であることから、＃７は＃８にタイムスタンプ値０’２（２３２７）を転送した（２２０３の動作）。一方、＃８ではデータ２３２２を出力し、データ２３２２のタイムスタンプ値は０４であることから＃８は＃７にタイムスタンプ値０’４（２３１５）を転送した（２２０３の動作）。

【００８１】

そして、＃７は＃１～＃３から送信された計算データを保持するキュー２３４０～２３４２の先頭データのタイムスタンプ値を比較し、最も古いデータ（２３１４）を出力データとする（２２０７の動作）。一方、＃８は＃４～＃６に対応するキュー２３４３～２３４５の先頭データのタイムスタンプ値を比較し、最も古いデータ（２３１８）を出力データとする（２２０７の動作）。

【００８２】

そして、＃７では、出力データ２３１４のタイムスタンプ値 ０３と、＃８から受け取ったタイムスタンプ値 ０’４（２３１５）に許容時刻 １０を足した値を比較する（２３１２）。そして、出力データ２３１４のタイムスタンプ値のほうが古かったので出力データ２３２０をクライアントに転送する（２２０４、２２０６の動作）。そして、出力データ２３２０のタイムスタンプ値 ０３を＃８に転送する（２２０５の動作）。

【００８３】

一方、＃８では、出力データ２３１８のタイムスタンプ値 １’５と、＃７から受け取ったタイムスタンプ値（２３２７）に許容時刻 １０を足した値を比較する（２３１９）。そして、出力データ２３１８のタイムスタンプ値のほうが新しかったので出力データ２３１８はクライアントに出力されない（２２０４の動作）。

【００８４】

次に、本実施例における、実行ノード出力制御方式の動作を述べる。図２４は実行ノード出力制御方式のフローチャートである。実行ノードでは第１実施例と同様にオペレータを処理する。そして、オペレータ処理完了後（２４０１）、出力ノードに計算データを送信する（２４０５）。その時に、実行ノードに共有キューが存在するノードでは（２４０２）、第１実施例とは異なる動作を取る。共有キューが存在する実行ノードでは複数の出力ノードへ転送する。その時に、共有キューでは複数の出力ノードへ転送するデータを保持する。共有キューを持つノードは、デー出力時に計算データのタイムスタンプ値が，他のキューの先頭データのタイムスタンプ値に許容時間を加算した時刻よりも古ければ出力ノードにデータを送信し（２４０３）、新しければデータを送信せずキューにデータを保留する（２４０４）。そして、出力ノードに保留中と通告する（２４０４）。そして、データを受信した（２４０６）出力ノードでは、他の実行ノードが保留中でなければ（２４０７）、実行ノードから送信された最新データの時刻を比較し最古のデータを出力する（２４０８）。

【００８５】

図２５はクエリ２１０１を用いた実行ノード出力制御方式の動作例である。図２３と同様に、＃１～６（２３０２～２３０７）は実行ノードであり、＃７,８（２３０８、２３０９）は出力ノードである。＃１～３、＃４～６で処理されたデータはそれぞれ＃７、＃８に転送される。例えば、＃３では銘柄 Ａ電のデータ（２５１０）が処理され、＃７に転送され（２５１７）クライアントに出力される（２３２１）。また、＃４では共有キュー２５１２を持ち、銘柄 Ｈ電、Ｂ銀のデータ（２５１３、２５１４）を処理する。処理された計算データは、＃８に転送されクライアントに出力する（２３２２）。また、同時に＃７に計算データのタイムスタンプ値を転送する。

【００８６】

今、＃４で処理された計算データ２５１４を共有キュー２５１２に投入した。その時、＃４では、＃７へ転送されるデータを保持するキューの先頭データ２５１３のタイムスタンプ値と、#8へ転送されるデータを保持するキューの先頭データ２５１４のタイムスタンプ値に許容時刻ずれ幅 １０を足した値を比較する（２５１１）。そして、２５１３のタイムスタンプ値のほうが古かったので＃７にタイムスタンプ値２５１８を転送する。＃７では＃１～＃３からの計算データを保持するキュー（２３４０～２３４２）の先頭データ（２５１７）とタイムスタンプ値２５１８を比較する。そして、データ２５１７が最古データなので、クライアントに出力する（２３２０）。

【００８７】

また、#４では、計算データ２５１４を共有キュー２５１２に投入した時に、＃８へ転送されるデータを保持するキューの先頭データ２５１４のタイムスタンプ値と、＃８へ転送されるデータを保持するキューの先頭データ２５１３のタイムスタンプ値に許容時刻ずれ幅 １０を足した値を比較する（２５１５）。そして、２５１４のタイムスタンプ値のほうが新しかったので、＃８に保留中（２３４３）と通告し、＃８からは、＃５、＃６からの計算データを保持するキュー（２３４４、２３４５）のデータは出力されない。

【実施例３】

【００８８】

最後に、タイムスタンプ付加を複数のタイムスタンプ付加ノードで行う、第３の実施例を説明する。図２６は第３の実施例の動作を説明するための図である。第１及び第２の実施例においては、説明は省略したが１つのノードでタイムスタンプ付加の処理をしていたため、ノード数が増加した場合にはタイムスタンプ付加処理が過負荷になりスループットが一定以上向上しない可能性があった。そこで、本実施例では、タイムスタンプ付加を第一のノードに追加された複数のタイムスタンプ付加ノードで処理することでスループットをさらに向上させる。

【００８９】

第３実施例では、図２６に示すようにストリームデータ処理サーバ２０６において、ストリームデータ処理エンジンにタイムスタンプ付加部２６０５、２６０６を含むタイムス

タンプ付加ノード２６０３、２６０４が第一のノードの集合として追加される。また、メインノード２０７のストリームデータ処理エンジンにタイムスタンプ付加ノード管理部２６０２がある。

【００９０】

動作としては、まず、メインノード２０７で読み込まれたユーザ定義クエリ２０３に対し、タイムスタンプ付加処理を分散するよう指定されていたならば、クエリ並列化部２０９で、クエリ２０３のタイムスタンプ付加処理を複数のノードで行うためのデータの振分け方を解析する。そしてデータの振分け方法からタイムスタンプ付加ノード参照テーブル２６０１を生成する。タイムスタンプ付加ノード参照テーブル２６０１は、データのレコード名をインデックスとしてそのデータのタイムスタンプを付加するノード名を参照するテーブルである。

【００９１】

そして、クエリの実行時に、入力データ２０２がメインノード２０７に取り込まれた時に、そのタイムスタンプ付加ノード管理部２６０２でタイムスタンプ付加ノード参照テーブル２６０１を参照し、入力データ２０２のレコード名からデータのタイムスタンプを付加するノード名を検索し、タイムスタンプ付加ノード２６０３、２６０４に入力データを割当てる。そして、タイムスタンプ付加ノード２６０３、２６０４のタイムスタンプ付加部２６０５、２６０６でそれぞれデータにタイムスタンプを付加し、実行ノード２３４、２３５に転送する。転送後、第１及び第２実施例と同様にオペレータを処理する。

【００９２】

第３実施例では、第１実施例でタイムスタンプ付加処理がボトルネックになっている場合にスループットを向上させる。例えば、第１実施例で、実行ノード数４、出力ノード数２で処理する時(図１２)、メインノードにおける処理可能な入力レートをＴ（タプル／秒）とすると、第１実施例では、全体で処理可能な入力レートがＴ（タプル／秒）になる(一つの処理・出力ノードの処理可能な入力レートがＴ／２(タプル／秒)以上の場合)。一方、第３実施例で、タイムスタンプ付加ノード数２で処理する時（図２９）、全体で処理可能な入力レートは２Ｔ（タプル／秒）となる(一つの処理・出力ノードの処理可能な入力レートがＴ／２(タプル／秒)以上の場合)。

【００９３】

第３実施例の動作詳細を述べる。図２７はクエリの登録時及び実行時のフローチャートである。

【００９４】

最初に、図２７上段に基づき、クエリ登録時の動作を述べる。クエリにはあらかじめタイムスタンプ付加を複数ノードで行うか否かについて印付けされている。印付けは、新たな構文を用いても、設定ファイルに記述しても、その他の方法を用いてもよい。そして、タイムスタンプ付加の分散が指定されていたならば、まず、第１実施例と同様に出力処理におけるデータの振分け方法を解析（２７０１）後、同じノードで出力処理をするデータは必ず同じノードでタイムスタンプの付加をように、タイムスタンプ付加処理のデータの振分け方法を解析する（２７０２）。そして、タイムスタンプ付加処理のデータの振分け方からタイムスタンプ付加ノード参照テーブルを生成しメインノードに配信する（２７０３）。

【００９５】

次に、第２７図下段によりクエリ実行時の動作を述べる。まず、クライアントから入力されたデータをメインノードが受信し（２７０５）、タイムスタンプ付加ノード参照テーブルにより、入力データのレコード名からタイムスタンプ付加するノード名を検索し、対応するタイムスタンプ付加ノードに入力データを送信する（２７０６）。そして、各タイムスタンプ付加ノードでは受信したデータにタイムスタンプを付加し（２７０７）、実行ノードにタイムスタンプを付加したデータを送信する（２７０８）。その後、実行ノードでは、受信したデータに対し第１実施例と同様にオペレータの処理をする。

【００９６】

図２８はクエリ３０１を用いたクエリ登録時の動作例である。まず、第１実施例と同様に出力処理におけるデータの振分け方としてカラム 銘柄によって振分ける方法２８０１を決める（２７０１の動作）。そして、出力処理を同じノードでするデータは必ずタイムスタンプの付加を同じノードでするように、２８０２のように出力処理と同様にカラム銘柄によるデータ振分け方法に決定する（２７０２の動作）。そして、タイムスタンプ付加処理のデータの振分け方からタイムスタンプ付加ノード参照テーブル３５０９を生成する（２７０３の動作）。最後に、第１実施例と同様に生成した実行木や出力ノード参照テーブルと共にタイムスタンプ付加ノード参照テーブル３５０９を該当するノードに転送する（２７０３の動作）。

【００９７】

図２９は、クエリ３０１を用いたクエリ実行時の動作例である。まず、メインノード９０２でタイムスタンプ付加ノード参照テーブル２９０１を用いて、＃１（９０７）、＃２（９１０）にそれぞれ銘柄％２が０であるデータ（銘柄 Ａ電（２９０３））、銘柄％２が１であるデータ（銘柄 Ｂ銀、Ｈ電のデータ（２９０２））を振り分ける（２７０６の動作）。そして、＃１、＃２でそれぞれ振り分けられたデータにタイムスタンプを付加し（２９０４、２９０６）（２７０７の動作）、付加されたデータに対して、第１実施例と同様にそれぞれ実行木２９０５、２９０７を処理し、処理されたデータを出力クライアントに転送する。

【００９８】

第３実施例ではサーバ上のメインノードでデータのタイムスタンプ付加ノードへの振分けをする。しかし、サーバでなくクライアントによって入力データをあらかじめ振り分けることも可能である。ストリームデータ処理においては入力先のクライアントが多数存在することもあり、そのような場合にはサーバによる振分けと比較して新たに振分けるためのノードを追加することなしに、スループットを向上させることができ有用である。入力クライアントによる振分け動作について図３０のフローチャートを用いて説明する。

【００９９】

まず、同図上段に示すクエリの実行開始時の動作を述べる。メインノードではクエリ登録時にタイムスタンプ付加ノード参照テーブルが格納される。まず、入力クライアントはメインノードに入力ストリーム名を送信する（３００１）。メインノードは、受信した入力ストリームに対応するタイムスタンプ付加ノード参照テーブルを入力クライアントに送信する（３００２）。そして、入力クライアントは受信したタイムスタンプ付加ノード参照テーブルを入力データ送信部に格納する（３００３）。

【０１００】

次に、同図下段に示すクエリ実行時の動作を述べる。まず、入力クライアントは、データ入力時に、実行開始時に取得したタイムスタンプ付加ノード参照テーブルから入力データの振り分け先を検索し（３００４）、各タイムスタンプ付加ノードに入力データを送信する（３００５）。そして、各タイムスタンプ付加ノードでは受信データに対してタイムスタンプを付加する。以降の動作は、第３実施例に従う。

【０１０１】

入力クライアントによる割当て方式の動作例についてクエリ３０１を用いて説明する。図３１はクエリ実行開始及び実行時の動作例を示す図である。

【０１０２】

まず、同図上段のクエリ実行開始時の動作例を説明する。まず、入力クライアント １．２．３．５（ＩＰアドレス）（３１０１）は入力ストリーム名 株注文（３１０３）をメインノード #0（９０２）に転送する（３００１の動作）。そして、#0では株注文に対応するタイムスタンプ付加ノード参照テーブル９０６を１．２．３．５に転送する（３００２の動作）。

【０１０３】

そして、同図下段に示すクエリ実行時に、入力クライアントではタイムスタンプ付加ノード参照テーブル９０６を参照し、入力データを振り分ける。すなわち、銘柄％２が０で

あるデータ（３１１１）はタイムスタンプを付加するノード#1（３１０７）に、銘柄％２が１であるデータ（３１１０）は、同じくタイムスタンプを付加するノード＃２（３１０８）に割当てられる（３００４、３００５の動作）。そして、#1、#2では、それぞれ銘柄％２が０であるデータ（３１１２）、銘柄％２が１であるデータ（３１１３）にタイムスタンプを付加しオペレータを処理する（３００６の動作）。

【産業上の利用可能性】

【０１０４】

本発明は、ストリームデータ処理システムにおいて、データごとに複数ノードでオペレータ処理や出力処理をする技術として有用である。

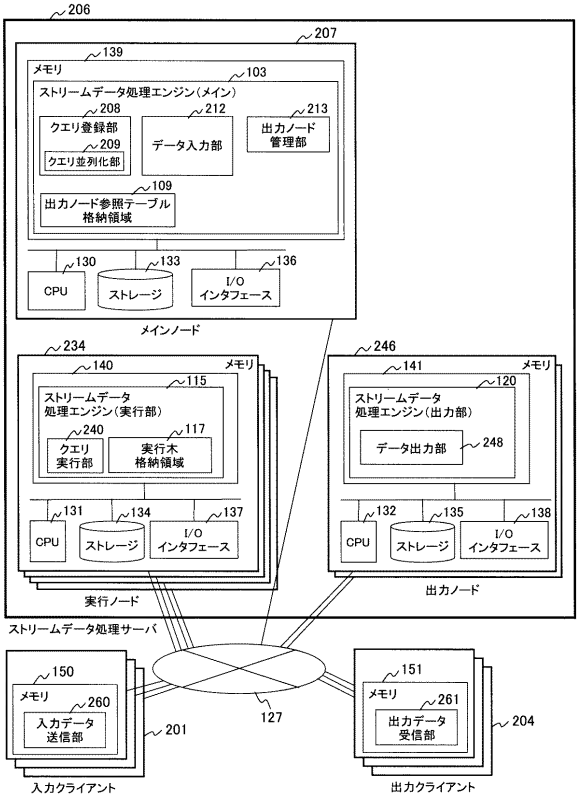
【符号の説明】

【０１０５】

１０３…ストリームデータ処理エンジン（メイン）
１０９…出力ノード参照テーブル格納領域
１１０…実行ノード参照テーブル格納領域
１１５…ストリームデータ処理エンジン（実行部）
１１７…実行木格納領域
１２０…ストリームデータ処理エンジン（出力部）
１２７…ネットワーク
１３０、１３１、１３２…ＣＰＵ
１３３、１３４、１３５…ストレージ
１３６、１３７、１３８…Ｉ／Ｏインタフェース
１３９、１４０、１４１、１５０、１５１　メモリ
２０１…入力クライアント
２０２…入力データ
２０３…順序保証グループ指定ユーザ定義クエリ
２０４…出力クライアント
２０５…クライアント名、出力レコード名
２０７…メインノード
２０８…クエリ登録部
２０９…クエリ並列化部
２１１…出力ノード参照テーブル
２６０…実行ノード参照テーブル
２１２…データ入力部
２１３…出力ノード管理部
２１６…ストリームデータ処理サーバ
２３４、２３５、２３６…実行ノード
２４６、２４７…出力ノード
２３７、２３８、２３９…実行木
２４０、２４１、２４２…クエリ実行部
２４３、２４４、２４５…計算データ
２４８、２４９…データ出力部
２５０、２５１…クライアント名、出力レコード名
２５２、２５３…出力データ
２６０…力データ送信部
２６１…出力データ受信部
１９０１、１９０２、２００１…出力時刻調整部
２６０１…タイムスタンプ付加ノード参照テーブル
２６０２…タイムスタンプ付加ノード管理部
２６０３、２６０４…タイムスタンプ付加ノード
２６０５、２６０６…タイムスタンプ付加部。

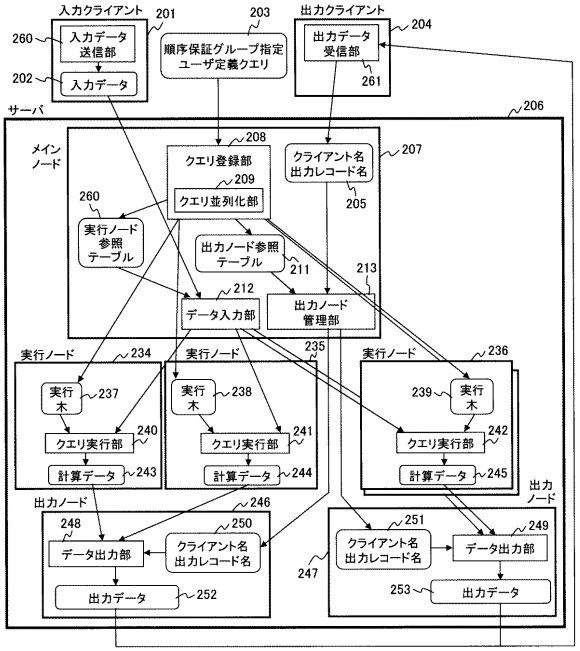
【図１】

図 1



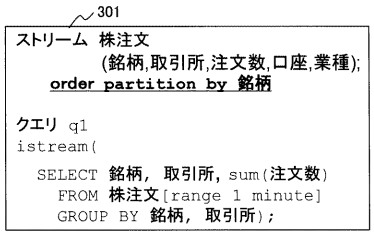
【図２】

図 2



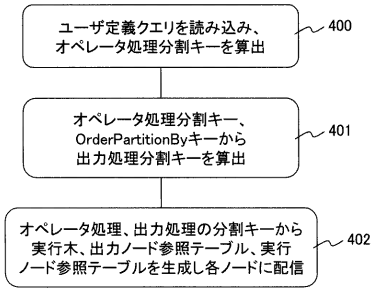
【図３】

図 3



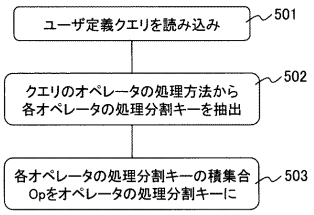
【図４】

図 4



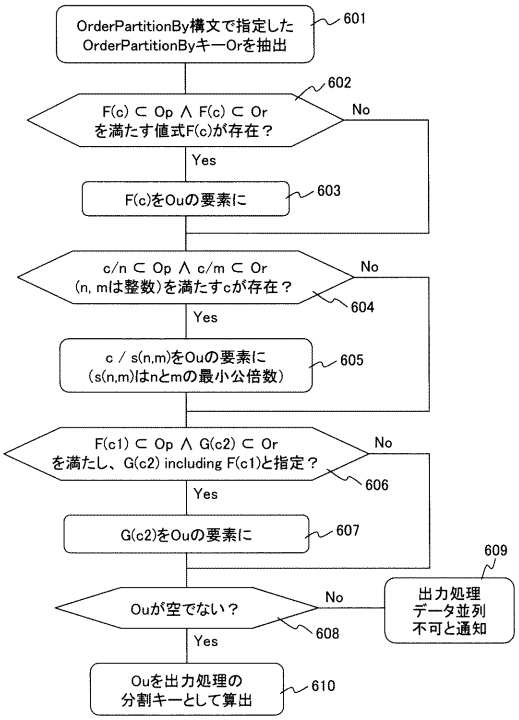
【図５】

図 5



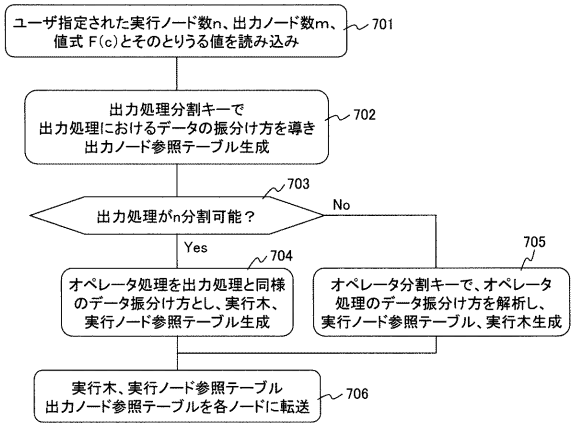
【図６】

図 6



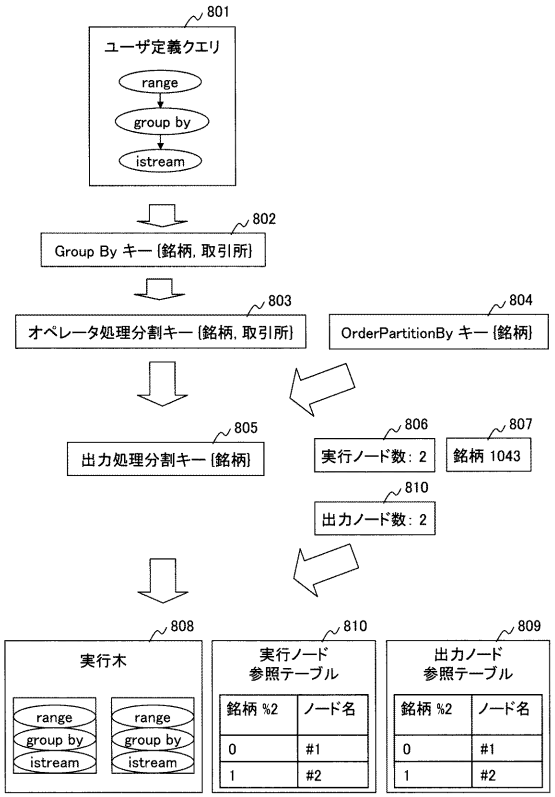
【図７】

図 7



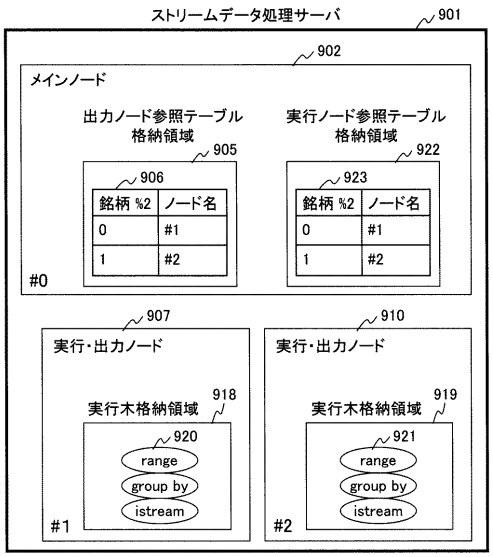
【図８】

図 8



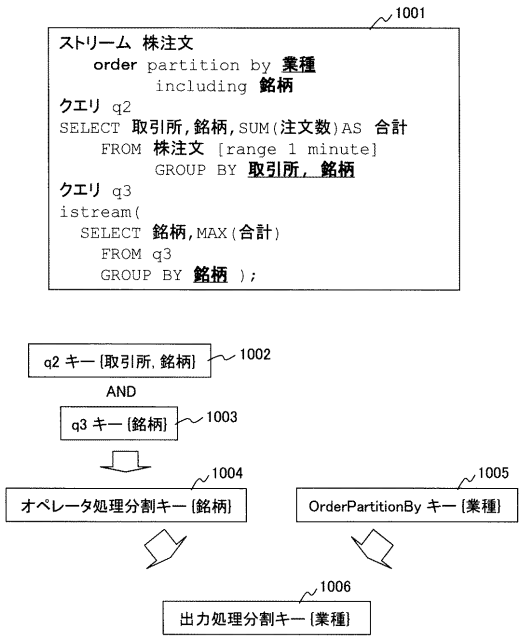
【図９】

図 9



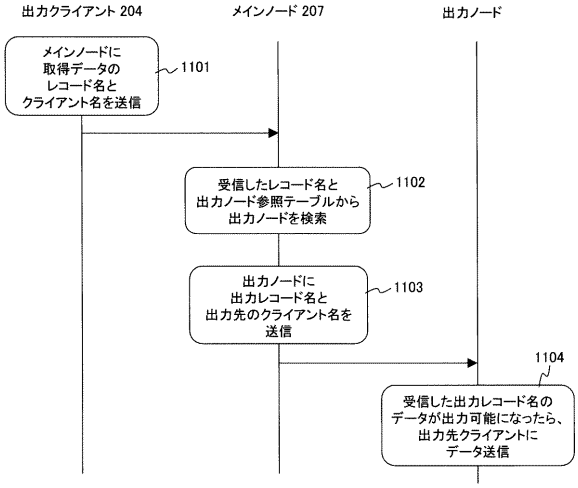
【図１０】

図 10



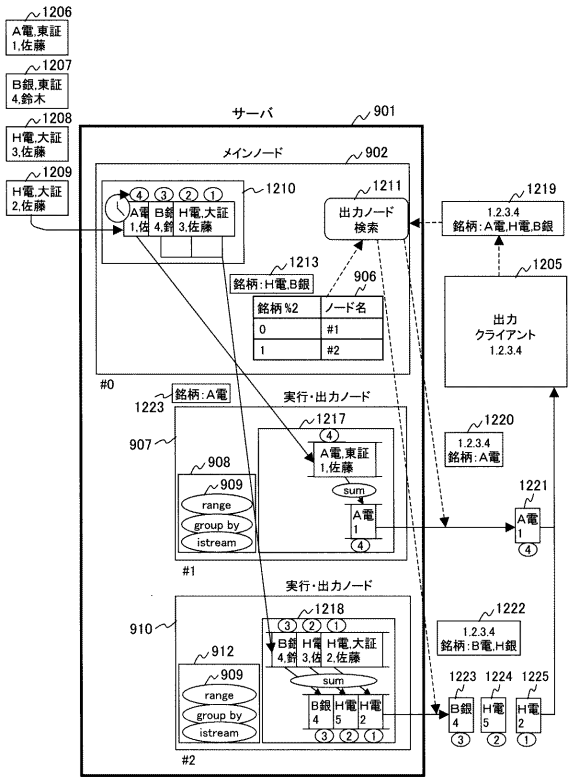
【図１１】

図 11



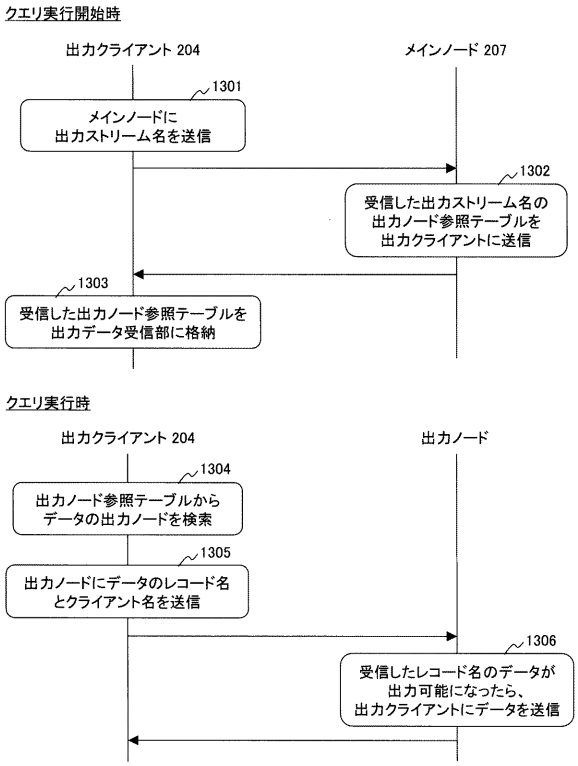
【図１２】

図 12



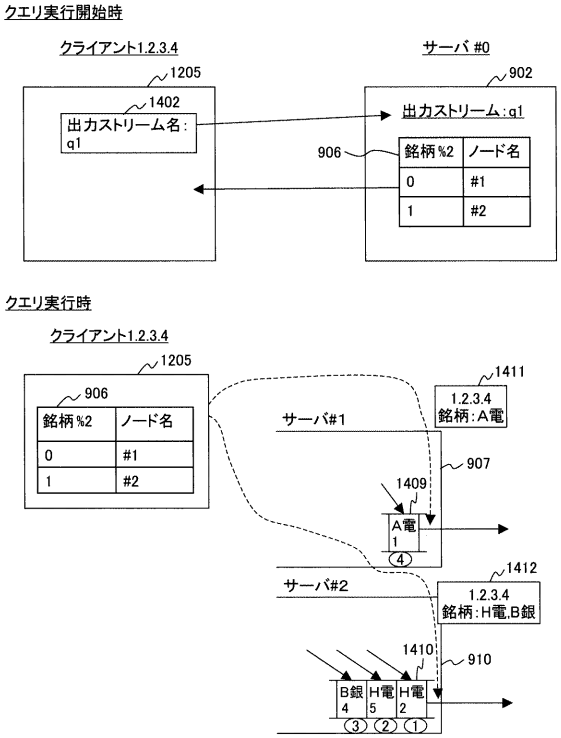
【図１３】

図 13



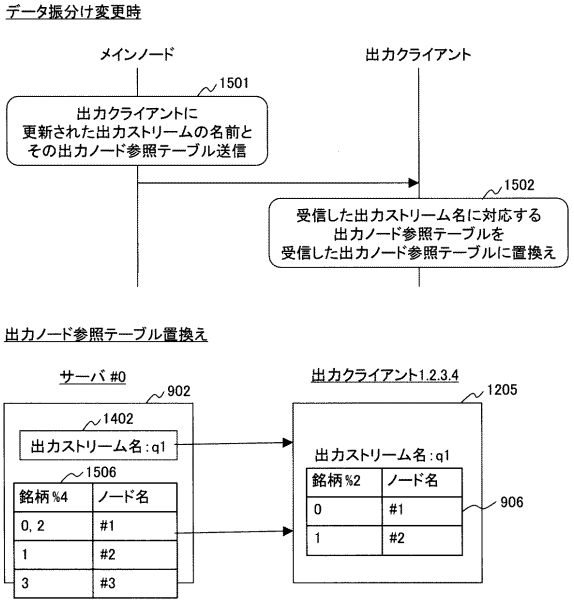
【図１４】

図 14



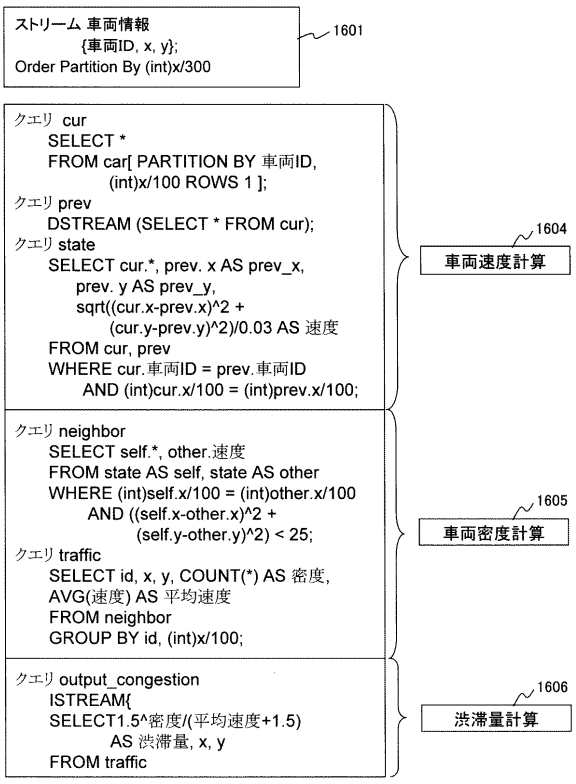
【図１５】

図 15



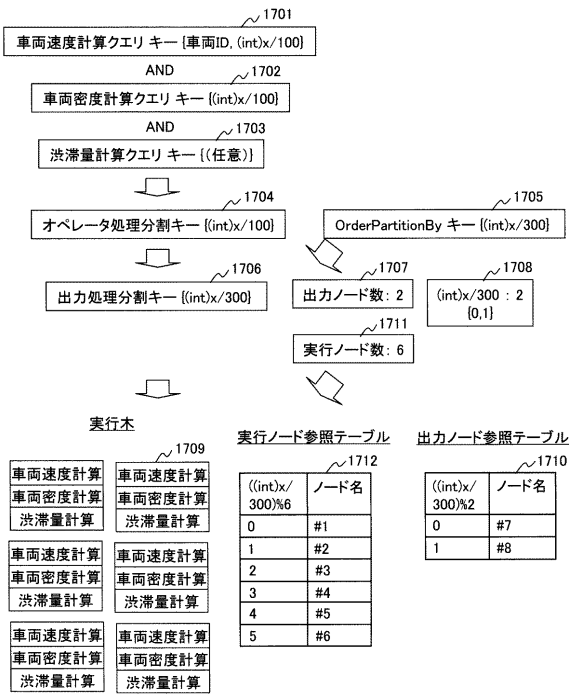
【図１６】

図 16



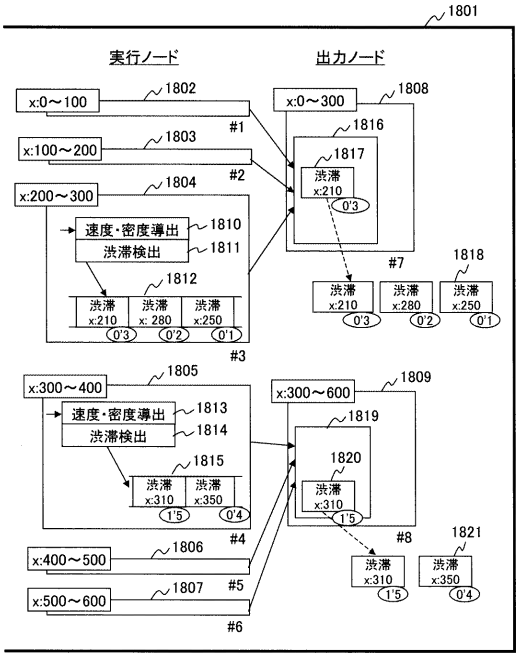
【図１７】

図 17



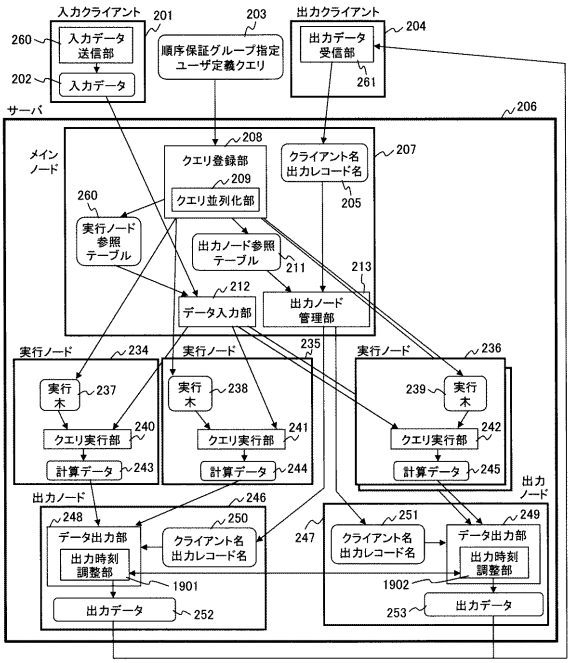
【図１８】

図 18



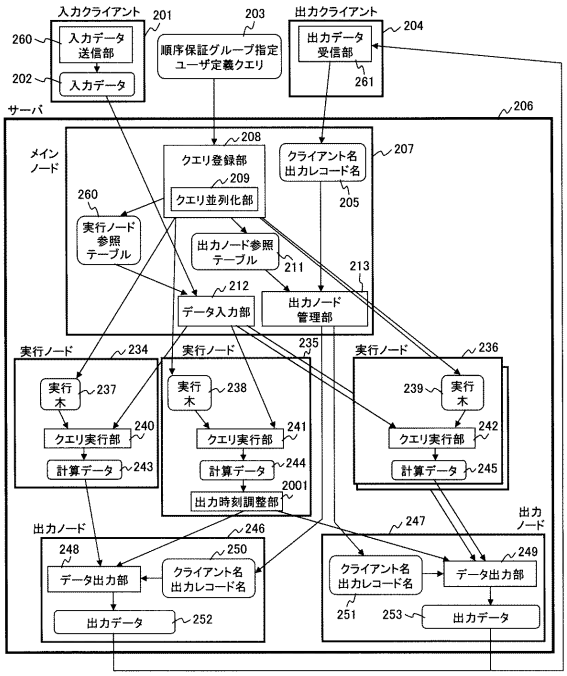
【図１９】

図 19



【図２０】

図 20



【図２１】

図 21

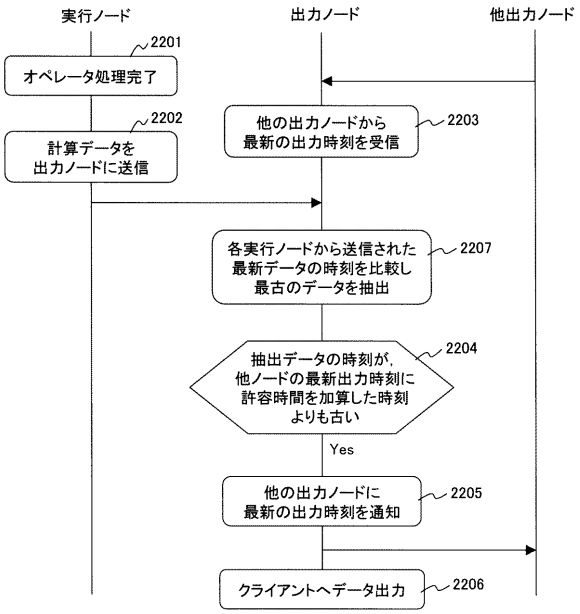
2101

```
ストリーム 株注文
        (銘柄,取引所,注文数,口座,業種);
    order partition by 銘柄
        limit 1 second

クエリ q1
istream(
  SELECT 銘柄, 取引所, sum(注文数)
    FROM 株注文[range 1 minute]
    GROUP BY 銘柄, 取引所);
```

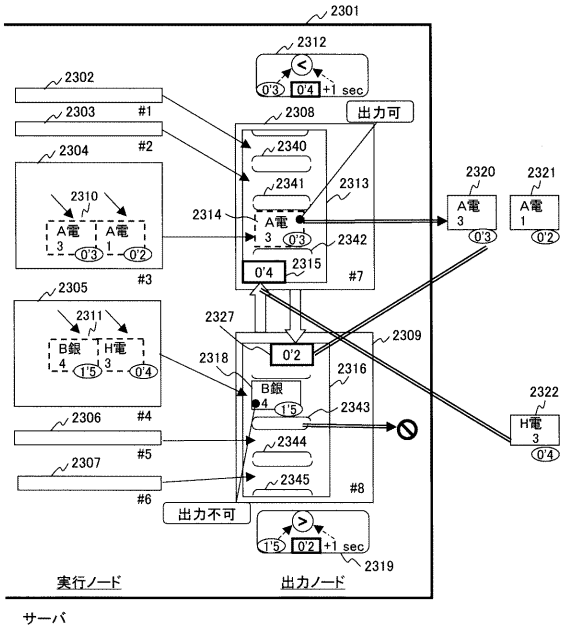
【図２２】

図 22



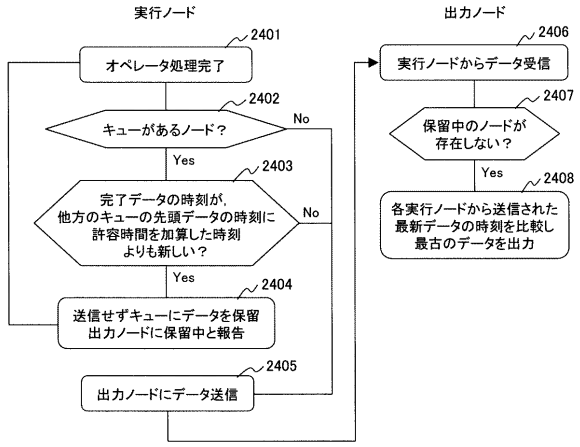
【図２３】

図 23



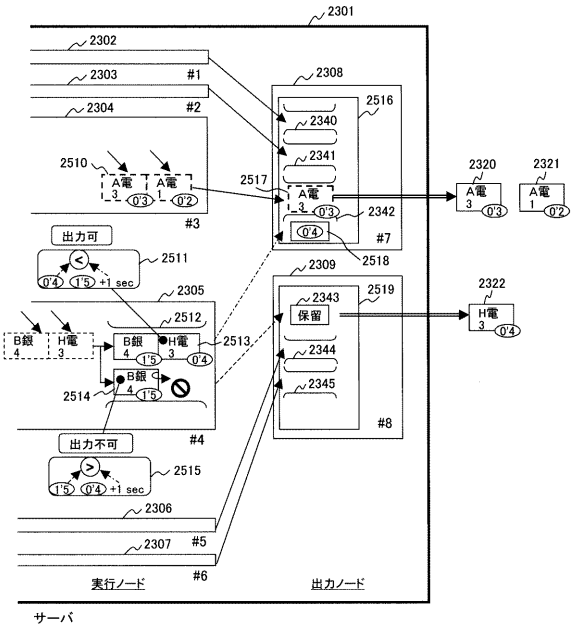
【図２４】

図 24



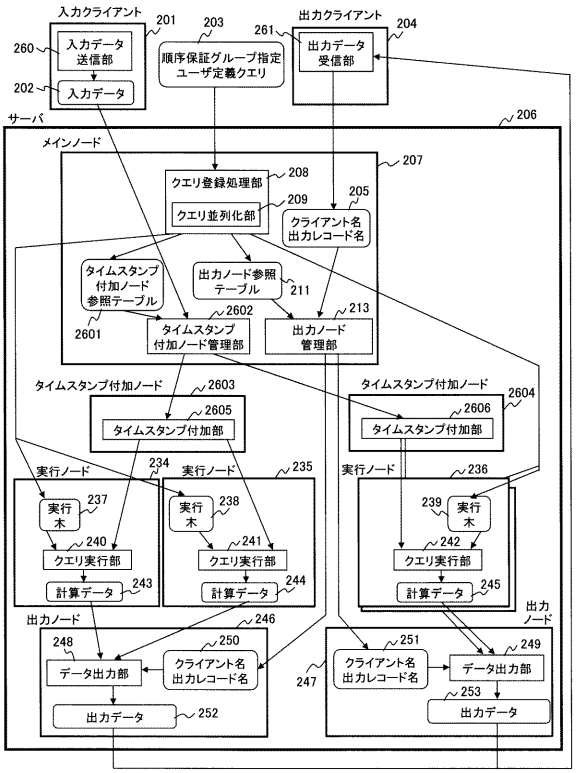
【図２５】

図 25



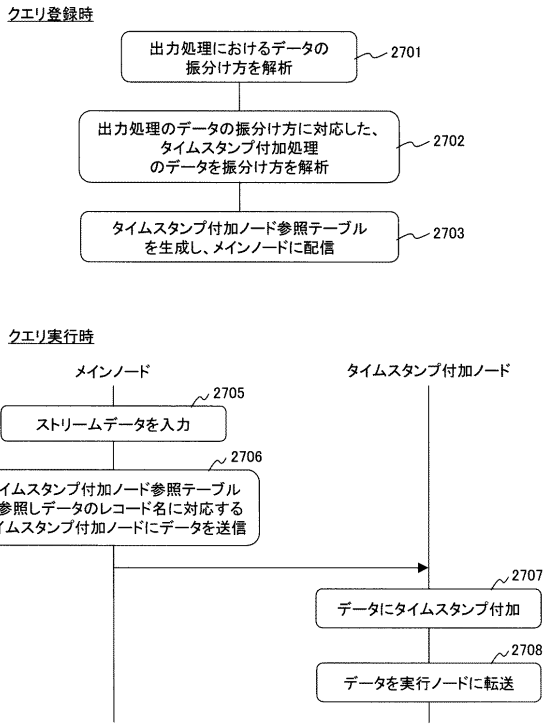
【図２６】

図 26



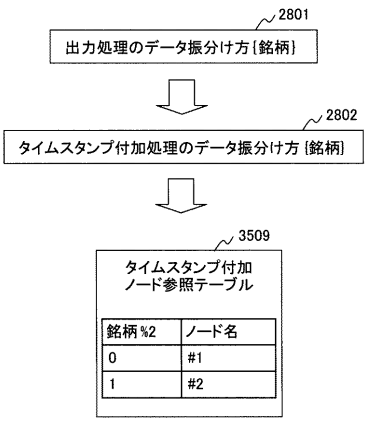
【図２７】

図 27



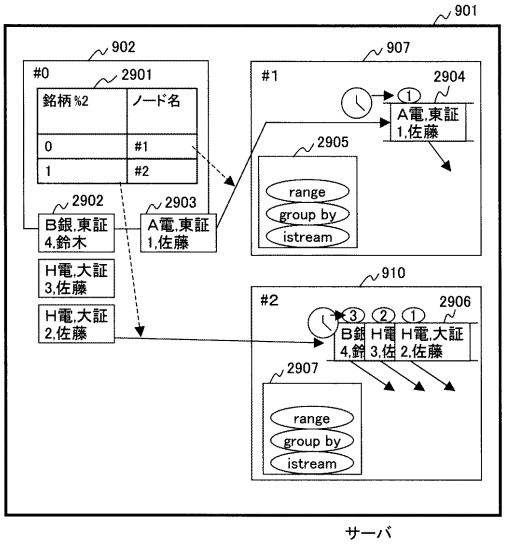
【図２８】

図 28



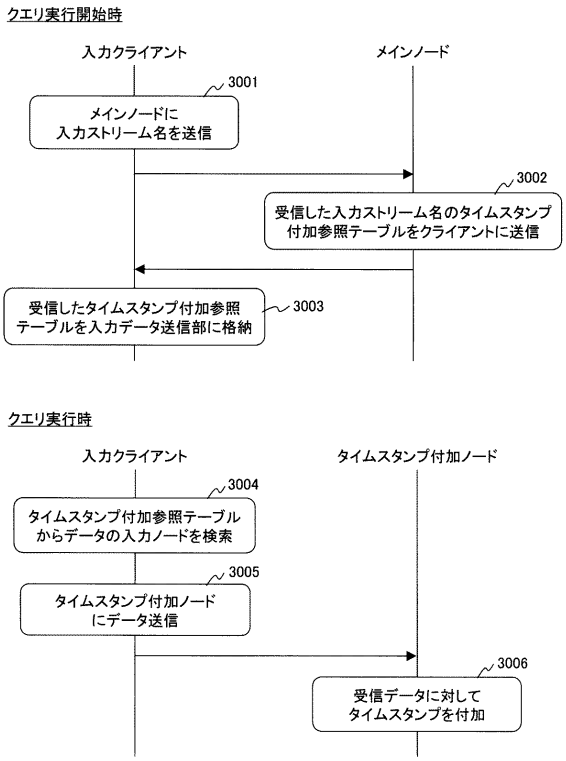
【図２９】

図 29



【図３０】

図 30



【図３１】

図 31