ブロックチェーンプラットフォームと汎用トランザクション管理システム

フォン ヤオカイ
馮　堯 鍇[1],　　櫻井　幸一[2]

概要：ブロックチェーン技術は，大きなアプリケーションもできることは既に分かって，その産業的価値が確認されつつある．ブロックチェーンはトランザクション管理システムに進化しているため，多くの企業や政府機関は，企業グレードのデータベースを置き換える，または補完するためにブロックチェーンを応用・検討している．本稿では，ブロックチェーン技術とデータベース技術との融合の現状からブロックチェーンデータベースシステムの開発例および業者の動きまで述べる．

キーワード：　ブロックチェーン，トランザクション，データベースシステム，PostgreSQL

Blockchain Platform and General Transaction Management System

Yaokai Feng[1],　Kouichi Sakurai[2]

Abstract: The blockchain technology has already been confirmed to have great industrial values, knowing that it can be used for many large-scale applications. Blockchain platforms have come to support general-purpose transactions. Such applications have expanded to the IoT, intelligent manufacturing, supply chain management, data storage and general purpose transactions. This paper describes the current state of the fusion of blockchain technology and database technology, as well as an example of the development of blockchain database systems.

Keywords: Blockchain, Transaction, Database system, PostgreSQL

1 ブロックチェーンの基本

1.1 仕組み

ブロックチェーンは暗号通貨ビットコインシステムの実現技術として誕生されたものである．具体的には，2008 年に記事・論文[1]が公開し，2009 年に初の実装コードを公開した．ブロックチェーンでは，履歴トランザクションレコードが連続的に積み重ねられたブロックチェーンが形成され，チェーン上のブロックが変更されたらそのブロックのハッシュ値が変わり，後続のブロックのハッシュ値にも影響がでるため，改ざんすることは非常に困難で，従来のトランザクションモードでデータの偽造などを効果的に解決できる．ブロックチェーンは，中央集権的な役割の排除および悪意ある参加者の存在（ビザンチン障害）が考慮され，信頼を前提せず分散システムを運営するための基盤である[2]．

ブロックチェーンシステムは，パブリックチェーン（public/permissionless blockchain，誰も参加可能のビットコインやイーサリアムなど）と，プライベートチェーン（private/permissioned blockchain，メンバーシップ管理がある Hyperledger Fabric [3]など）に分類できる[4]． パブリックブロックチェーンでは，誰も参加できるため，高価な PoW コンセンサスを使用している． それに，各トランザクションはシリアル的に実行される．結果として，トランザクションの処理能力が制限される． 一方，プライベートブロックチェーン（例えば，Hyperledger Fabric）では，並列トランザクション処理をサポートする [3]．しかし，各利用者は信頼できるメンバーシップサービスに登録する必要がある．サプライチェーン，ヘルスケア，リソース共有などのアプリケーションはプライベートチェーンを利用する．

1.2 ブロックチェーンの応用状況

ブロックチェーンスキームに基づいて，ビットコインは約 20 年経って，大きな脆弱性なく，数万の分散ノードで中断なく

[1] 九州大学大学院システム情報科学研究院
Kyushu University, Fukuoka 819-0395, Japan
fengyk@ait.kyushu-u.ac.jp
[2] 九州大学大学院システム情報科学研究院
Kyushu University, Fukuoka 819-0395, Japan
sakurai@inf.kyushu-u.ac.jp

稼働している．ビットコインシステムの後ろにあるブロックチェーン技術は，大きなアプリケーションもできることは徐々に認識され，その産業的価値が確認されつつある[5]． 共通の信頼できる第三者に依存することなく，ピアツーピア（P2P）環境で動作する最初のデジタル通貨（または暗号通貨）は単純なトランザクション（ビットコインの移動）のみを実行できるが，Ethereum [6]や Hyperledge[3]などのブロックチェーンプラットフォームは汎用トランザクションをサポートするようになった．ブロックチェーンはトランザクション管理システムに進化しているため，データの透明性と障害に対するセキュリティも含め，分散データベースに匹敵する．多くの企業や政府機関は，企業グレードのデータベースを置き換える，または補完するためにブロックチェーンを検討している[7,8,9]．

ブロックチェーンのアプリケーションは，金融の始まりからモノのインターネット IoT，インテリジェント製造，サプライチェーン管理，データストレージ，トランザクションに拡張された．その他の分野は，クラウドコンピューティング，ビッグデータ，ベアラネットワークなどの次世代情報技術の開発に新たな機会をもたらす．信頼性の高いメカニズムは，現在のソーシャルビジネスモデルを変え，それによって技術革新と産業変化の新しいラウンドをトリガーする．多数の組織はセキュリティ，コンセンサス，データベース，ベンチマーク，検証など，さまざまな側面に取り組んでいる．

グローバル関連のテクノロジー市場は 2024 年までに 77 億 4,000 万ドルに達すると言われている[10]．

1.3 ブロックチェーンプラットフォームの例

IMB 社が BaaS (Backend as a Service)の１つとして，開発した，ひろく使われている Hyperledger Fabric[5]を一例としてブロックチェーンプラットフォームを説明する．Hyperledger Fabric は，複数機関のビジネスネットワークを完全に統合するエンタープライズ対応の開発，ガバナンス，および運用のためにビジネスネットワークを民主的に管理するツールである[10,11,12]．2015 年 8 月にプロトタイプ Open Blockchain を公開し始めてから，2019 年 11 月には Hyperledger Fabric v1.44 を公開した．

ブロックチェーンアプリケーションを迅速に構築する開発者ツールとして，Hyperledger Fabric は元帳を提供し，一連の直観的な操作ツールで管理するプライベートブロックチェーンシステムで，メンバーのアイデンティティーと役割が互いに既知となるプライベートな許可制ビジネス・ネットワークを対象としたフレームワークである．また，セキュリティー，スケーラビリティー，機密性を備えたネットワークを実現し，柔軟な展開オプションをサポートしている．以下のコンポーネントからなる[12]．

（ア）**資産：**キーと値のペアの集合として表現される．

（イ）**共有レジャー：**資産の状態と所有権を記録するもので，ワールド・ステートとブロックチェーンとの 2 つのコンポーネントから構成される．ワールド・ステートは特定の時点でのレジャーの状態の記述で，レジャーのデータベースである．ブロックチェーンはすべてのトランザクションを記録するトランザクションのログ履歴である．

（ウ）**スマート・コントラクト：**資産および関連するトランザクションを定義するソフトウェア（チェーン・コードとも呼ばれる）として，ブロックチェーン内のすべてのノードによって実行されるユーザー定義の計算であるスマートコントラクトである[2]． システムのビジネス・ロジックを格納するものとも言える．アプリケーションがレジャーとやり取りする必要な時に呼び出される．チェーン・コードを作成するには，主に Golang または Node.js という言語を使用する．

（エ）**ピア・ノード**：ピアはブロックチェーン・ネットワークのメンバーである．

（オ）**チャネル：**ピアのグループごとに個別のトランザクションのレジャーを作成することが可能になる．

（カ）**組織：**ピアはさまざまな組織に属する．各組織はメンバーシップ・サービス・プロバイダーを利用して各ピアにアイデンティティー（デジタル証明書）を割り当てる．

（キ）**メンバーシップ・サービス・プロバイダー（MSP）：**メンバーのアイデンティティーと役割の認証局である．

（ク）**順序付けサービス**：複数のトランザクションを順序付けて，ブロックにパッケージ化した後，チャネル上のピアに配信する．

汎用ブロックチェーンではアプリ開発者の視点からのデータの保存先はブロックチェーンそのものではなく，ワールドステート（キー・バリュー・ストア）であり，ワールドステートのハッシュ値をブロックチェーンに書き込む[2]．ブロックチェーンは「履歴」の管理に適したデータ構造で，データベースのように複雑なモデルの大容量のデータの場合，データそのものは外部に置き，そのデータの権利の転移情報などブロックチェーンの特性が生きる部分だけをブロックチェーンで管理するのが良い[2]．

Hyperledger Fabric ネットワークがバックエンドとして，フロントエンドのアプリケーションのネットワークと通信する．SDK を使用してクライアント・アプリケーションを開発する．ユーザーのチェーン・コードを実行し，ネットワーク内でトランザクション，イベントのモニターなどを行う [12]．

2 ブロックチェーン技術とデータベース技術の融合

研究[24]では，データベースとブロックチェーンという 2 つのテクノロジー間の相互融合の最新状況をまとめた．具体的には，データベーステクノロジーからブロックチェーン技術の開発への影響，およびブロックチェーンテクノロジーからいくつかのデータベースでの新しい機能の導入への影響である．

ブロックチェーンは，分散化，チェーンハッシュを使用した暗号化セキュリティ，管理制御なし，不変性，中央機関の許可なしに転送する自由など，多くの点で従来のデータベースとは異なる．しかし，近年，この２つの技術はかなり融合して来た．多くのエンタープライズアプリケーションはブロックチェーンを使用して従来のデータベースストレージソリューションをアップグレードした．基本的に，ブロックチェーンには従来

のデータベースにあるいくつかの機能がまだない．ブロックチェーンは，従来のデータベースをブロックチェーンと統合するか，ブロックチェーン指向の分散データベースを作成することにより，従来のデータベース機能を活用できる．データベース機能を含めることで，低遅延，高スループット，高速スケーラビリティ，およびブロックチェーンデータに対する複雑なクエリを備えたシステムができる．したがって，ブロックチェーンとデータベースの両方の機能を備えたアプリケーションは，効率とセキュリティの両方も強化される．現在，ブロックチェーンプラットフォームの多くもデータベースと統合され，多くのブロックチェーンデータベースが開発され導入されている．

2.1 ブロックチェーンとデータベースとの主な異なるポイント

研究[13]では，ブロックチェーンとデータベースのプロパティが異なる重要なポイントとして，以下を列挙した．どちらも互いの特性を活用および強化できる．

(ア) 通常，ブロックチェーンのトランザクションのレイテンシは，データベースのより高い．従って，データベースを使用することで，待ち時間を必要に応じて低くすることができる．

(イ) ブロックチェーンのトランザクションには，シリアライズ可能な分離が必要である．これは，強い整合性を提供するコンセンサスアルゴリズムによって実現できる．データベースには，2フェーズロックおよび同時実行制メカニズムがある．ただし，MongoDBに基づくBlockchainDBなどの新しいブロックチェーンデータベースは，ブロックチェーンに基づく新しいトランザクションメカニズムを提供し始めた．

(ウ) ほとんどのブロックチェーンプラットフォームは履歴データに対する複雑なクエリをサポートしていない．複雑なクエリ機能はほとんどのデータベースで使用できる．

(エ) ブロックチェーンの分散機能により，過去10年以上の間にほとんどの金融システムと業界が影響を受けた．中心なしの分散化は，従来の分散データベースでは利用できない．新しいブロックチェーンスタイルのデータベースの出現により，中心なしの分散化が可能になり，多くのアプリケーションで使用される有望な成長をリードしている．

(オ) ブロックチェーンにより，デジタルアセットの作成と移動が可能になる．これは，従来のデータベースでは許可されていない．ただし，ブロックチェーンスタイルの分散データベースでは，この機能を組み込み機能として使用できる．

2.2 既存のデータベースシステムに基づいたブロックチェーンプラットフォーム

データベースシステムは，ブロックチェーンのトランザクションデータを保存するために使用された．既存の多くのデータベースの異なる特性に基づいて，特定のブロックチェーンアプリケーションで使用するデータベースを選択できる．研究[24]では，既存のデータベースシステムのブロックチェーンシステムによる利用される状況をまとめた．

(ア) PostgreSQL．さまざまなネイティブデータタイプがあり，ユーザー定義オブジェクトをサポートする．これは，ブロックチェーンシステムでブロックチェーンアセットを定義するのに役立つ．高度にモジュール化されており，拡張可能であり，さまざまなレベルでの分離もサポートする．PostgreSQL はブロックチェーンリレーショナルデータベースの作成に使用されており，レプリカは相互に信頼していないさまざまな組織によって管理されている．具体的な実現方法は次の章で詳しく説明する．

(イ) MariaDB． MySQL派生として開発されている，オープンソースのDBMSで，高度なレプリケーション機能とクラスタリング機能を備えたオープンソースのリレーショナルデータベースシステムである． OurSQL は，MariaDBシステムのサーバーとしてのOurSQLはブロックチェーンとMariaDBの組み合わせで，プライベートブロックチェーンアプリケーションに使用できる．

(ウ) SQLite．組み込みの非Client-sever式で，ACID準拠のリレーショナルデータベースシステムである．ローカルデータベースとしてブロックチェーンノードに埋め込むのに適していると言われている．

(エ) CovenantSQL（CQL）．分散型で信頼できるGDPRに準拠しており，SQLite に基づいて構築されたブロックチェーン機能を備えている．低コストのサービスとしてのデータベース（DBaaS）として使用できる． CQLは，グローバルコンセンサスレイヤー，SQLコンセンサスレイヤー，データストアレイヤーで構成される階層型アーキテクチャである．

他に，Dqlite 1（分散SQLite）はインメモリオプションを備えた，オープンソースの高速なディスクバックアップデータベースで，フォールトトレラントなIoTおよびEdgeデバイスに最適である． RQLITE はオープンソースの軽量でフォールトトレラントな分散リレーショナルデータベースで，ノードのクラスターの動的な作成を可能にし，ノード間暗号化を提供する．RQLITE は軽量ブロックチェーンソリューションの潜在的な候補と考えられる．

以上はリレーショナルデータベースシステムである．他に，研究[24]では，NoSQL データベースシステム（MongoDB, RethinkDB, CouchDB, QLDB, TiesDB, CosmosDB, HBase 等）および NewSQL データベースシステム（VoltDB, TiDB, CockroachDB 等）のブロックチェーンシステム化の状況も紹介した．例えば，RethinkDBに基づいてBigchainDBは構築された；MongoDB にブロックチェーンを対応できるレイヤーを追加してProvenDBを構築できた．

3 ブロックチェーンデータベースの実現例

節3.2で言及したPostgreSQLに基づくブロックチェーンプラットフォームの実現は，本章で詳しく紹介する．

3.1 ブロックチェーンデータベースシステムのメリット

ブロックチェーンプラットフォームとリレーショナルデータベースは概念的には異なるが，論文[14]では両者の提供する機能間のいくつかの類似点を紹介する上で，データベースでよく知られている概念の観点からブロックチェーン技術を説明・

再構築することができると強調した．ブロックチェーンデータベースシステムは最初から構築するのではなく，既存のリレーショナルデータベースの機能を活用して，ブロックチェーンプラットフォームを構築できるのを示した．具体的に言えば，実現したブロックチェーンデータベースシステムは，既存のブロックチェーンによって提供されるすべての機能を備えた．複雑なデータ型，制約，トリガー，複雑なクエリなども支持する． さらに，ブロックチェーンアプリケーションの構築をデータベースアプリケーションの構築と同じくらい簡単にできる．豊富な分析クエリおよび強力なコンプライアンスと監査が必要である金融サービスおよびサプライチェーンなどのアプリケーションはブロックチェーンデータベースから最も恩恵を受ける可能性が高い．

例えば，節2.1で紹介したHyperledger Fabricプラットフォームはブロックチェーン台帳の状態を管理する機能であるスマートコントラクトをサポートし，スマートコントラクト関数を呼び出すことによってトランザクションを実現する[16]．これは，リレーショナルデータベースのストアドPL/SQLプロシージャに似ている．具体的には，両方とも入力引数を受け入れ，executing によってデータを取得し，定義済みロジックでデータベースへのデータを書き戻す．一方，ブロックチェーン元帳は，攻撃者がブロックを改ざんして検出を回避することが難しいのに対して，伝統データベースにはすべてのトランザクション，実行されたクエリ，ユーザー情報が記録されるが，攻撃による変更は検出し辛い．

ブロックチェーンデータベースのメリットは次の3点が挙げられる．

1）数十年にわたる研究開発ですでにリレーショナルデータベースに組み込まれている豊富な機能とトランザクション処理機能を活用することにより，既存のブロックチェーンプラットフォームが提供するすべての機能を備え，複雑なデータ型をより良くサポートするブロックチェーンリレーショナルデータベースを構築することができる [14]．

2）バックアップ，分析，および既存のエンタープライズシステムとの統合に利用できる豊富なツールを活用できる．

3）ブロックチェーンデータベースを利用すれば，ブロックチェーンアプリケーションの構築をデータベースアプリケーションの構築と同じくらい簡単にできる．

信頼できないデータベースノードが独立してトランザクションを実行し，同じシリアル化可能な順序でそれらをコミットすることを保証する必要がある．これを達成するために，コンセンサスアルゴリズムを利用してブロック間の順序を決める．そして，SSI (Serializable Snapshot Isolation)で各ブロックのトランザクションのコミット順序の一致性を確保する．

論文[16]では PostgreSQL を利用してブロックチェーンデータベースを実現した．

3.2 ブロックチェーンシステムとデータベースシステムとの比較および必要な拡張

論文[14]では，８つの機能におけてブロックチェーンシステムとデータベースシステムを比較することによって，ブロックチェーンデータベースを構築するために必要な拡張を明示する．

1) Smart contract & transaction: Hyperledger Fabricなど多くのブロックチェーンプラットフォームは，関数であるスマートコントラクトをサポートする．取引スマートコントラクト関数の呼び出しによってブロックチェーン台帳の状態を管理する．リレーショナルデータベース内のストアド PL / SQL プロシージャはこれと似ている．どちらも入力引数を受け入れ，実行してデータを取得する．ただし，Random 或いは timestamp を利用する場合は，PL / SQL プロシージャの実行結果には不確定性がある．したがって，PL / SQL を制約する必要がある．

2) User authenticity & non-repudiability: プライベートブロックチェーンシステムは，ユーザー管理と信頼性の確保のために公開鍵インフラストラクチャを採用している．組織に所属する参加ユーザーは，ブロックチェーンネットワークへの参加を許可されている．トランザクションは呼び出し側によってデジタル署名され，ブロックチェーンに記録され，それらを否認不可にする．リレーショナルデータベースには，ユーザー，ロール，グループ，およびさまざまなユーザー認証オプションを利用する高度なユーザー管理機能がある． ただし，提出および記録されたトランザクションは呼び出し側によって署名されず，否認可能になる． したがって，トランザクションには呼び出し元の署名が必要である．

3) Confidentiality & access control:一部のプライベートブロックチェーンは，スマートコントラクト，トランザクション，およびデータに許可されたユーザーのみがアクセスできる機密性の概念をサポートしている． さらに，関数の呼び出しとデータの変更へのアクセスは，特定のユーザーに制限されている． リレーショナルデータベースは，テーブル，行，列，およびPL / SQL プロシージャのアクセス制御を包括的にサポートしている． 一部のリレーショナルデータベースは，コンテンツベースのアクセス制御，暗号化，データの分離などの機能さえ提供する．

4) Immutability of transactions and ledger state: ブロックチェーン台帳は，ブロックの追加専用ログで，ブロックの改ざんおよび検出の回避は困難である．データベースでは，送信されたすべてのトランザクション，実行されたクエリ，ユーザー情報が記録されるが，デジタル署名がないため，ログへの変更を検出することはできない．

5) Consistent replicated ledger across distrustful nodes: ブロックチェーン内の障害のないすべてのピアは，すべてのトランザクションと元帳の状態の一貫したレプリカを維持する必要がある． これは，合意を通じて合意されたのと同じ順序でトランザクションをコミットするすべてのピアによって保証される． データベースは，マスタースレーブおよびマスターマスタープロトコルを使用して，ノード間のトランザクションログと状態のレプリケーションをサポートする．これらのプロトコルでは，マスターノードは，トランザクションを実行して最終的な更新を他のマスターとスレーブに伝播する唯一のノードであるため，信頼されている． したがって，レプリカの整合

性を維持しながら，分散信頼の概念を備えたレプリケーションプロトコルを提案する必要がある．

6) Serializability isolation, ACID: ブロックチェーン・トランザクションにはシリアル化可能な分離が必要である．この場合，dirty read，繰り返し不可の読み取り，phantom read，およびシリアル化の異常は不可能である．トランザクションを並行して実行する場合，すべてのノードで同じシリアライズ可能な順序に従う必要がある．さらに，トランザクションはACIDに準拠している必要がある．(1) 厳密な2フェーズロック，(2) オプティミスティック同時実行制御，または(3)SSI を使用することで，データベースでシリアル化可能な分離を実現できる．これは，コンセンサスによって決定されたブロック順序に従うように拡張する必要がある．

7) Asynchronous transactions: トランザクション処理とコンセンサスには重要な遅延が伴う場合があるため，クライアントはトランザクションを非同期に送信し，通知メカニズムを利用してトランザクションが正常にコミットされたかどうかを確認する．データベースは，アプリケーションが利用できる通知チャネルとトリガーのサポートを提供する．

8) Provenance: ブロックチェーン内のトランザクションの監査可能な追加のみのログは，サプライチェーンの追跡や財務コンプライアンスを含むいくつかのユースケースの provenance store として活用できる．ただし，今日のほとんどのブロックチェーンプラットフォームは，履歴データに対する複雑なクエリをサポートしていないか，provenance queries 用に最適化されていない．スナップショット分離など，マルチバージョン同時実行制御をサポートする特定のリレーショナルデータベースでは，行のすべてのバージョンが維持され，古い行はオンデマンドまたは定期的に削除される．ただし，SQL クエリは古い行にアクセスできない．Provenance query の場合，パージを無効にし，古い行のクエリを有効にする必要がある．

3.3 何で PostgreSQL?

PostgreSQL [15]は，更新または削除後もすべての行を物理的に保存する．さらに，高度にモジュール化され，拡張可能である．これらの理由により，ゼロからデータベースを実装するのではなく，PostgreSQL を変更してブロックチェーンリレーショナルデータベースを構築する．実際の変更は約 4000 行の C コードを追加するだけである．

PostgreSQL で，リレーションの各行には，ヘッダーに 2 つの追加要素，xmin と xmax がある．これらは，それぞれ行を作成および削除したトランザクションの ID で，行に対するすべての更新は，削除とそれに続く挿入である．削除された行には，実際に削除されるのではなく，xmax を設定することでフラグが付けられる．これは，すべてのバージョンのデータを維持するブロックチェーンを構築するという目標に理想的である．

クライアントは，libpq などのアプリケーションインターフェイスを使用して，PostgreSQL サーバーに接続する．次に，クエリを実行するために，各クライアント接続にバックエンドプロセスが割り当てられる．

3.4 主要コンポーネント

本システムには次のコンポーネントがある．

(ア) クライアント (Client)：クライアントユーザーのネットワークへの登録が必要で，通知チャネルをリッスンしてトランザクションステータスを受け取ることもできる．

(イ) データベースピアノード (Database Peer Node)：一組織は 1 つまたは複数のノードを運営している．各データベースノードは，元帳の独自のレプリカをデータベースファイルとして保持つ．ストアドプロシージャとしてスマートコントラクトを独立して実行し，注文サービスによって形成されたトランザクションのブロックを検証およびコミットする．

(ウ) Ordering Service：信頼できないデータベースノードの間に合意が必要である．異なる信頼モデルを使用したコ

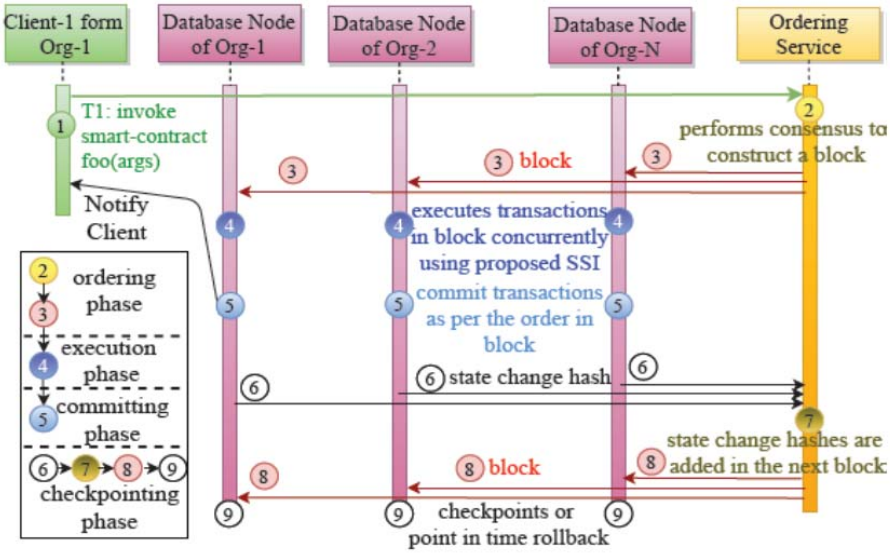

図１．トランザクションフロー(Order-then-execute)[14].

Fig.1. Transaction flow of order-then-execute) [14].

ンセンサスアルゴリズムは既に多く提案された．Ordering サービスはコンセンサスノードまたはordering ノードで構成され，それぞれが異なる組織に所有されている．コンセンサスの出力はトランザクションのブロックを生成し，すべてのデータベースノードにブロードキャストされる．

3.5 トランザクションフロー：Order-then-Execute

クライアントから Order-then-Execute アプローチで提出したトランザクションは (a) 一意の識別子，(b) クライアントのユーザー名，(c) PL / SQL プロシージャの実行プロシージャの名前と引数，(d) Hash(a, b, c)を利用するデジタル署名，という内容を含め，クライアントの秘密鍵を使用する． トランザクションフローは 4 つのパイプラインフェーズ：Ordering; Execuion; Committing and Check-pointing からなる．全体のフローは図１に示している．

１） Ordering phase:

クライアントはトランザクションを１つの Ordering service node に送信する．Ordering service node 同士の間は，定期的なタイムアウト（1 秒ごとなど）でコンセンサスプロトコルを開始し，トランザクションのブロックを構築する．各トランザクションにグローバル順序を付けてから，そのブロックをすべてのノードに配信する．図１の手順２と３はこのフェーズを示す．

２） Execution phase:

トランザクションのブロックを受信すると，各データベースノードは，受信したブロックをローカルファイルシステムに保持されているブロックストアに追加する．各ノードは並行して，ブロックシーケンス番号の順に未処理のブロックをブロックストアから一度に１つずつ取得し，次の３つの操作を実行する．

ア）データベースノードは，トランザクションごとにスレッドを割り当て，認証と実行を行う．同時に，元帳テーブルでは各トランザクションを記録する．

イ）各スレッドはトランザクションのユーザー名に関連付けられた公開キーを取得して，ユーザーのデジタル署名を検証してから，トランザクションごとに渡された引数を使用して PL / SQL プロシージャを実行する．図１の手順４はこのフェーズを示す．

ウ）各スレッドがトランザクションの実行を完了すると，トランザクションは（プロシージャの実行ロジックに従って）コミットまたはアボートする準備ができたが，続行せずに待機する． これは，各ノードの実行完了順序が異なる可能性があるからである．

３） Committing Phase:

コミット順序はすべてのデータベースノードで同じになるように，トランザクションのブロックに表示される順序とする．すべての有効なトランザクションが実行され，コミットまたはアボートの準備ができた場合にのみ先に進む．図１では，ステップ５はコミット段階を示している．すべてのトランザクションの実行が完了しない限り，コミットフェーズを開始できない．

４） Check-pointing Phase:

ブロック内のすべてのトランザクションが処理されると，各ノードは書き込みセットのハッシュを計算する．各ノードは後続のブロックを受信すると，他のノードによって計算された書き込みセットのハッシュを確認する．すべての非障害ノードによってハッシュ値確認が終わったら，ノードはチェックポイントを記録する．図１では，ステップ 6, 7, および 8 はチェックポイント設定フェーズを示している．

3.6 PostgreSQL の変更と拡張

本システムを実装するには，PostgreSQL への追加および変更するコンポーネントについて説明する．

3.6.1 追加するコンポーネント

追加するコンポーネントは以下の通りである．

（ア）Communication Middleware & Block Processor:
次の２つの新しいバックグラウンドワーカーを導入した．(1) 他のノードおよび orderer と通信し，トランザクション／ブロックを転送および受信するための通信ミドルウェア． (2) ブロックを処理するブロックプロセッサ．

（イ）Shared Memory Data Structures:
Order-then-execute を実現するために，共有メモリに次の 2 つのデータ構造を導入した．(1) TxMetadata: TxMetadata は，ブロックプロセッサとバックエンドの間の通信と同期を可能にする．ブロックプロセッサはこのデータ構造を使用してすべてのトランザクションの実行が終わったかをチェックする． (2) BlockProcessorMetadata は，トランザクションをコミット/アボートすると，バックエンドからブロックプロセッサのシグナリングに役立つ．さらに，このデータ構造は，最後にコミットされたブロック番号，現在のブロック番号，およびミドルウェアからブロックプロセッサへのシグナリングを有効にするセマフォを持っている．

（ウ）Blockchain Related Catalog Table:
２つのシステムカタログテーブル，pgLedger と pgCerts を導入した．pgLedger は節 3.6 で説明された各トランザクション情報を格納する元帳テーブル(ledger table)である．その情報は，グローバル識別子，ローカルトランザクション ID，トランザクションで渡されたクエリ，そのトランザクションの提出者（クライアント）およびコミット/アボートステータスを含める．pgCerts テーブルは，すべてのブロックチェーンユーザーの暗号化資格証明書を格納する．

（エ）Provenance Query:
この読み取り専用クエリタイプはアクティまたは非アクティブ（xmax でマークされている）に関係なく，コミットされたすべての行を見ることができる．すべての履歴コンテンツにアクセスできることによって，pgLedger を利用すれば，非常に複雑な析および監査クエリが可能になる．

3.6.2 変更するコンポーネント

Order-then-execute において，主に次の２つのコンポーネントを変更した.

(ア) Application Interface & Deterministic PL/SQL Procedures ブロックチェーントランザクションと Provenance query を提出することおよび各ノードで最新のブロックの高さを取得するができるように，デフォルトアプリケーションインターフェースの libpq を拡張した．つまり，PL / SQL プロシージャを確定的にするために，日付/時刻ライブラリ，数学ライブラリのランダム関数，シーケンス操作関数，およびシステム情報関数の使用を制限した． さらに，SELECT ステートメントでは，LIMIT または FETCH を使用するときに ORDER BY primary _key を指定する必要がある． さらに，WHERE 句で xmin，xmax などの行ヘッダーを使用できない.

(イ) SSI Based on Block Height 各行に２つのフィールドを追加しました：(i) 作成者ブロック番号，(ii) 削除者ブロック番号．コミット中，エントリが更新，挿入，削除のいずれであるかに応じて，TxWriteSet のエントリに対してこれら 2 つのフィールドが入力される． SI は，xmin と xmax を使用して行可視化ロジックを適用し，トランザクションで行を表示するかどうかを識別する．行の可視性ロジックを拡張して，行の作成者と削除者のブロック番号とトランザクションの snapshot-height を使用して追加の条件を設定する.

4 業者の動き

複数のブロックチェーンデータベースシステムは既に開発・公開されている．このようなプラットフォームを利用したら，大規模なエンタープライズアプリケーションから，ゲームまたは小規模プロジェクトまで，異なる分野のブロックチェーンと関連する分散型アプリケーション(DAPP)は似ている環境・方法で開発できる．数十年にわたる研究・開発してきたリレーショナルデータベースの豊富な機能（トランザクション処理機能，クエリなど）を活用することにより，既存のブロックチェーンプラットフォームが提供するすべての機能を備え，複雑なデータ型をより良くサポートするブロックチェーンリレーショナルデータベースを構築することができる．即ち，ブロックチェーンの特性を持つデータベースのようなものである．強力なクエリ機能，分散制御，不変データストレージ，アセットサポートなどは挙げられる．リレーショナルブロックチェーンに基づいたパブリックブロックチェーンである．つまり，SQL と同じくらい簡単にコーディングできることを意味する．開発者と企業は，スケーラブルなブロックチェーンデータベースを使用して，ブロックチェーンの概念実証，プラットフォーム，およびアプリケーションを展開できる．規模，セキュリティ，またはパフォーマンスを犠牲にすることなく，アイデンティティや知的財産からサプライチェーン，エネルギー，IoT，金融エコシステムに至る幅広い業界とユースケースをサポートする[11, 16, 17].

ブロックチェーン分野の先駆 ChromaWay 社，スタットアップの Bigchain 社および Microsoft や Oracle などの大手企業の動きを紹介する.

4.1 ChromaWay 社とスタットアップ会社 Bigchain のシステム

ブロックチェーン分野の先駆 ChromaWay 社に開発された Chromia[16] およびドイツのスタットアップ会社 Bigchain に開発された BigchainDB[11, 17]が挙げられる．Chromia は，SQL 言語らしい，もっと良い（もっとコンパクトなど）と言われている言語 Rell を使用する．Rell 言語の使用例を付録に示す.

4.2 Microsoft 社の Azure Blockchain Service

Azure Blockchain Service は完全に管理された台帳サービスであり，ユーザーは Azure でブロックチェーン ネットワークの拡張や操作を大規模に行うことができる[18]．Ethereum Quorum 台帳がサポートされている.次のような機能が提供される．(1) ネットワークの簡単なデプロイと運用；(2) 組み込みのコンソーシアム管理；(3) 使い慣れた開発ツールによるスマート コントラクトの開発

これらの機能は，ほとんど管理が必要なく，追加使用料もなしで利用できる．さらに，オープンソースのツールとプラットフォームを自由に選んでアプリケーションの開発を続けることができる[18].

4.3 Oracle 社の Blockchain Table

Blockchain Table は，Oracle Database の中で使える，データベーステーブルの新たな特殊なタイプである[17]．Blockchain Table では INSERT のみを行い，UPDATE などの修正は禁止である．削除にも制約がある．各行にはその前の行のハッシュを保持し，改ざんされていないことをユーザーは確認できる．また，オプションとして各行の内容に PKI ベースの電子署名で署名することができ，確実な否認防止を実現することもできている[19].

データへの署名および検証により，アプリケーション提供者による内部不正も防ぐことができる（「信頼するが検証もする」）．Oracle Database 内の Blockchain Table を活用することで，中央集権的アプリケーションを，非中央集権モデルへと変更することによるよりセキュアにすることができる[19]．例えば，金融取引の履歴，監査証跡，規制対応のコンプライアンスデータ，金融レコード，訴訟ホールドされたデータなどの証拠保全が必要な場合，および変更されないデータ（例えば IoT デバイスから受け取ったデータ，コンプライアンスデータなど）の場合では,，Blockchain Table の出番である．また，

１）この機能は Oracle Database の一部として提供されるため，新しい基盤の導入を必要としない.

２）SQL や PL/SQL，JDBC やその他のやり方でテーブルにアクセスできるので，利用は簡単である.

３）Blockchain Table はさらに，他のテーブルと組み合わせてトランザクションやクエリを行うことができる.

5 おわりに

ビットコインは約 20 年経って，大きな脆弱性なく，数万の分散ノードで中断なく稼働してきた．その後ろにあるブロックチェーン技術の産業的価値が確認されつつある．近年，データベース技術との融合によって，数十年にわたる研究開発で既にリレーショナルデータベースに組み込まれている豊富な機能とトランザクション処理機能を活用することができる．しかも，ブロックチェーンアプリケーションの構築をデータベースアプリケーションの構築と同じくらい簡単にできる．本稿では，ブロックチェーンの基本的な仕組みを紹介した後，ブロックチェーンおよびデータベース２分野の技術の融合の現状，開発例および大手業者の動きを紹介した．

謝辞：本研究は JSPS 科研費 JP18H03240 の助成を受けたものです．

参考文献

[1] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, 2008.

[2] NEC クラウドプラットフォーム事業部，データベースの視点から見るブロックチェーン：https://www.slideshare.net/Hyperledger_Tokyo/hyperledger-fabric (accessed on August 12, 2020).

[3] E. Androulaki, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. The Thirteenth EuroSys Conference, page 30. ACM, 2018.

[4] Pingcheng Ruan et al. “A Transactional Perspective on Execute-order-validate Blockchains”, arXiv:2003.10064v1 [cs.DC], 23 Mar 2020.

[5] P. Ruan, et al. Blockchains and Distributed Databases: a Twin Study. arXiv:1910.01310v1, 3 Oct. 2019.

[6] G. Wood. Ethereum: A secure decentralized generalised transaction ledger. Ethereum project yellow paper, 2014.

[7] V. Morabito. Business innovation through blockchain. Cham: Springer International Publishing, 2017.

[8] W. Mougayar. The business blockchain: promise, practice, and application of the next Internet technology. John Wiley & Sons, 2016.

[9] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, et al. Blockchain technology: Beyond bitcoin. Applied Innovation, 2(6-10):71, 2016.

[10] https://bitcoin.org/bitcoin.pdf (accessed on July 10, 2020).

[11] A New Era in Distributed Computing with Blockchains and Databases, C. Mohan, Singapore Management University (SMU), Singapore, 20 November 2018.

[12] https://www.ibm.com/developerworks/jp/cloud/library/cl-blockchain-hyperledger-fabric-hyperledger-composer-compared/index.html

[13] M. Raikwar, D. Gligoroski and G. Velinov, "Trends in Development of Databases and Blockchain," 2020 Seventh International Conference on Software Defined Systems (SDS), pp. 177-182, 2020. arXiv:2003.05687, 2020.

[14] S. Nathan, C. Govindarajan, A. Saraf, et al. Blockchain Meets Database: Design and Implementation of a Blockchain Relational Database. arXiv:1903.01919v2, 31 May 2019.

[15] RESEARCH BRIEFS, https://www.cbinsights.com/blog/bitcoin-blockchain-startup-market-map/ (accessed on August 10, 2020).

[16] https://chromia.com/index.html (accessed on August 19, 2020).

[17] https://www.bigchaindb.com/ (accessed on August 19, 2020).

[18] Azure Blockchain Service: https://docs.microsoft.com/ja-jp/azure/blockchain/service/overview (accessed on August 17, 2020).

[19] Oracle Blockchain Table: https://orablogs-jp.blogspot.com/2020/02/native-blockchain-tables.html (accessed on August 17, 2020).

付録 Rell 言語の使用例

ブロックチェーン分野の先駆 ChromaWay 社に開発されたブロックチェーンデータベースシステム Chromia のクエリ言語として Rell の使用例は次のものが挙げられる．

1) Create Statement：
create user(name = 'Bob', company = company @ { .name == 'Amazon' }); あるいは,
val name = 'Bob'; create user(name, company @ { company.name == 'Amazon' });

2) Update Statement：
update user @ { .name == 'Bob' } (company = 'Microsoft');
// Bob の company は”Microsoft”とする（１つだけ）
update user @* { .company.name == 'Bad Company' } (salary -= 1000);
//会社 Bad Company のすべてのユーザの salary を減額する．

3) Delete Statement:
delete user @ { .name == 'Bob' };
// 1 つのユーザを削除；
delete user @* { .company.name == 'Bad Company' };
//すべて削除

4) Query:
user @ { .name == 'Bill', .company == 'Microsoft' }
// returns a specific user (all conditions must match)