CBDCフォーラム
【新たなテクノロジーとCBDC】WG（WG4）
第12回会合 事務局説明資料

CBDCパイロット実験システムの概要

2025年12月22日

日本銀行 決済機構局

WG4での主な議論

データモデルに関する議論

口座残高型とUTXO（初回事務局資料P.14）

- 口座残高型の場合

現在の状態

利用者	残高
A	50
B	70
C	30

トランザクション(Tx)

操作
Aの残高を45減らす Bの残高を45増やす

(AがBに45円を送金)

Tx実行後の状態

利用者	残高
A	5
B	115
C	30

- 口座残高型以外、例えばUTXOの場合（※後述するProject Hamiltonでの考え方）

「システム内で未だ使用されていない」 Unspent Transaction Output=「現在使用可能な」価値情報
全UTXOを記録・更新するデータベース（UTXO set）で、デジタル通貨の流通を表現

現在の状態
(UTXO set)

価値	所有者
10	A
20	B
30	C
40	A
50	B

トランザクション(Tx)

費消 (入力)		追加 (出力)	
10	A	45	B
40	A	5	A

(AがBに45円を送金)

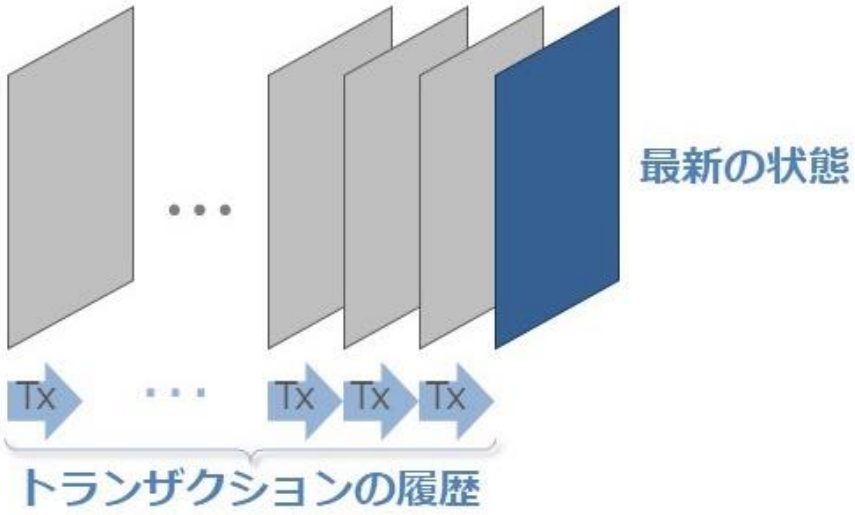
Tx実行後の状態

価値	所有者
20	B
30	C
50	B
45	B
5	A

データモデルに関する議論

状態の持ち方（初回事務局資料P.15）

- 伝統的な金融機関の勘定系システムのデータモデルでは、トランザクションの最新の状態を管理することを重視。
- 他方でイーサリアムやビットコインのデータモデルは、トランザクションの履歴を、ブロックチェーンで記録することを重視。



データモデルに関する議論

Tx処理集中による性能劣化は、Tx処理の粒度の調整で回避できる可能性

（参加者）２点質問がある。更新処理集中時の対応として、レコード分割を一案として挙げられているが、こうしたレコード分割を施した口座残高型は、ＵＴＸＯ型と整理できるのか。２点目は、ＵＴＸＯ型であっても更新処理集中の際にはレコードロックが必要となり、性能上のボトルネックとなり得るのではないか。

（プレゼンタ）１点目については、ＵＴＸＯ型をどう定義するか次第ではあるものの、レコード分割は口座残高型における性能上の工夫策の一つと理解しており、レコード分割を施した口座残高型は引き続き口座残高型と考える。２点目については、ＵＴＸＯ型においても更新処理が集中した場合には、後続処理を待機されるレコードロックは必要になると考えるが、トークンの粒度が細かいほど、更新処理の集中自体が起こりづらくなり、レコードロックに伴う処理遅延も発生しにくくなると考える。

（日本銀行）トークンの粒度の問題というのはその通りと考える。ＵＴＸＯ型と口座残高型は違う概念として論じられることが多いが、口座残高型でも口座を分けたりレコードを分割したりすれば、レコードロックの範囲を限定でき、実態としてＵＴＸＯ型と同様の並列処理が可能となって、処理集中に伴う性能面での問題が生じにくくなる、という見方もある。

（参加者）更新処理集中とそれに伴う性能劣化をどの程度回避できるかは、データモデルというより、ＵＴＸＯ型ならトークンの粒度の細かさ、口座残高型ならレコード分割の程度といったように、トランザクションの処理の粒度を最適化する方向で対応できる可能性があると理解した。

（出所）CBDCフォーラムWG4 第3回会合議事概要より抜粋

データモデルに関する議論

集計処理をどこまで行うかで、データモデル間の並列処理性の差は変化

比較

JPX

○> △> ▲（相対評価、▲もその一点をもって選択肢から除外するような致命的な欠点とまでは言えない）

		中央管理 口座残高型	中央管理 固定額面 トークン型	中央管理 UTXO型	コメント
性能	スループット	○	▲	△	送金時に更新対象となるレコード数はデータモデルごとに異なる（大まかな傾向としては、口座残高型が最も少なく、次がUTXO型、固定額面トークン型は最も多くなりそう）。このため、スループット要件を達成する際に使用するマシンパワーに違いが生じると考えられる。
	レイテンシ	○	▲	△	固定額面トークン型・UTXO型ではユーザ単位の残高計算が必要な分、レイテンシが悪化。 「両替」「お釣り」が必要な固定額面トークン型は相対的にレイテンシが悪いと考察。
	並列処理性	○	○	○	固定額面トークン型やUTXO型は複数の相手と同時に取引できそうに一見思えるが、各種制限判定のためにユーザ単位で排他制御を行うのであれば、モデル間の差異とならないと考察。

7

（出所）CBDCフォーラムWG4 第５回会合資料（株式会社JPX総研様）より抜粋

データモデルに関する議論

Tx処理の粒度を上げた場合の課題

WG4　第3回　NTTD発表内容サマリー（1/3）

1. 背景

中央銀行デジタル通貨に関する実証実験「概念実証フェーズ2」結果報告書に記載されている性能面に関する課題を取り上げ、各課題に対する対応の方向性を示した。具体的には、更新処理集中の緩和策、トランザクション保証の仕組み、DBサーバの水平スケーリング、口座残高型とUTXO型のプロコン分析である。

2. 更新処理集中の緩和策

設計パターン2（口座残高型）では、仲介機関を跨って送金する場合、仕向仲介機関のユーザ口と被仕向仲介機関のユーザ口の更新が必要となり、ユーザ口に更新処理が集中すると待ちが発生し、ボトルネックとなる。これに対して、ユーザ口のレコードを分割することで、分割した数だけ並列処理が可能となり処理集中を緩和できるが、残高合計値が必要な場合には足し上げる処理が必要になったり、処理が偏らないように分割したり、運用中にレコード分割の見直しが必要になるなど、追加の課題が生じる可能性がある。

この課題の対応の方向性として、レコードを分割せずに、複数のトランザクションを1つに集約する（ただし、取引の明細は1件ずつ出力する）ことで更新処理数を削減し、処理集中の影響を大幅に緩和する対策を提案した。

3. トランザクション保証の仕組み

DBの水平分割（複数のサーバ間で整合性のとれた処理の実現）や中央銀行台帳上の口座振替と仲介機関台帳上の口座振替を別々のタイミングで行う（取引のアトミック性の維持）場合には、システム内のトランザクション保証およびCBDCシステム全体（中央システム、仲介機関NW、仲介機関システム）でトランザクションを保証する仕組みが必要である。

特にCBDCシステム全体でトランザクションを保証するためには、仲介機関NWにおいて、トランザクションの処理状況の可視化、業務チェックエラー時の補償トランザクション（ロールバック）の実行、システムエラー時のバックアウトや再実行処理等が必要となる。

NTT DATA

（出所）CBDCフォーラムWG4 第3回会合資料（株式会社NTTデータ様）より抜粋

データモデルに関する議論

Tx処理の粒度を上げた場合の課題

比較（つづき）

JPX

○＞△＞▲（相対評価、▲もその一点をもって選択肢から除外するような致命的な欠点とまでは言えない）

		中央管理 口座残高型	中央管理 固定額面 トークン型	中央管理 UTXO型	コメント
信頼性	セキュリティリスクへの耐性	○	○	○	いずれのモデルでも、中央銀行や仲介機関におけるセキュリティリスク管理は必要。モデル間に差異は見当たらない。
	取引のアトミック性（整合性）	○	○	○	いずれのモデルでも、取引ごとにアトミック性を確保することが可能。モデル間に差異は見当たらない。
拡張性	追加サービスとの親和性	▲？	○？	△？	「お金に色を付ける」ようなユースケースがあれば、最も親和性が高いのは固定額面トークン型で次点でUTXO型か（口座残高型でも工夫次第かもしれず「？」つき）。
	実装容易性	○	△～▲	△	構成がシンプルでマシンパワーの消費も少ない口座残高型が相対的に実装容易と推察。 固定額面トークン型は、送金時に指定するトークンを選択するアルゴリズム次第で複雑さが増す（実装難度が上がる）可能性。

8

（出所）CBDCフォーラムWG4 第５回会合資料（株式会社JPX総研様）より抜粋

DLTの活用可能性

台帳は中央集権DBで、それ以外でDLTを組み合わせるハイブリッドモデル

DLTと中央集権システムをハイブリッドに融合

SORAMITSU ソラミツ

4

・ハイブリッドシステムにより、　「状態を更新」と「状態の変化の蓄積」を両立
・DLTに基づく情報の改ざん防止と中央集権DBによるスケールメリットの両者を享受
・デジタル人民元は、二重使用の防止・改ざん防止に一部DLT技術を使用と報道

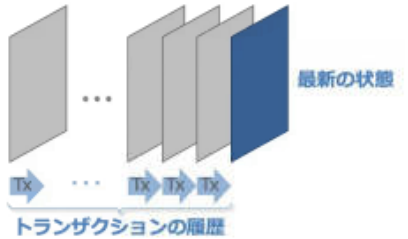
1. 残高管理を中央集権DBで実施
   ・高スループット/低レイテンシを実現
   ・スケーラブルキューなどによるスケールアウト

2. 「取引履歴」をDLTに蓄積
   ・「取引履歴」を「状態の変化」として蓄積
   ・「取引履歴」の改ざん防止・真正性の担保
   ・不正な送金時に「取引履歴」を活用して検証

3 両方の利点を活用したハイブリッド・システム
   ・分散化された「残高台帳」で、即座に「状態を更新」
   ・「取引履歴」をキューに書き込む
   ・その後、「取引履歴」を不変な「状態の変化の蓄積」としてキューからDLTに記録

分類の軸②：「状態を更新」するか「状態の変化を蓄積」するか

- 伝統的な金融機関の勘定系システムのデータモデルでは、トランザクションの最新の状態を管理することを重視。
- 他方でイーサリアムやビットコインのデータモデルは、トランザクションの履歴を、ブロックチェーンで記録することを重視。



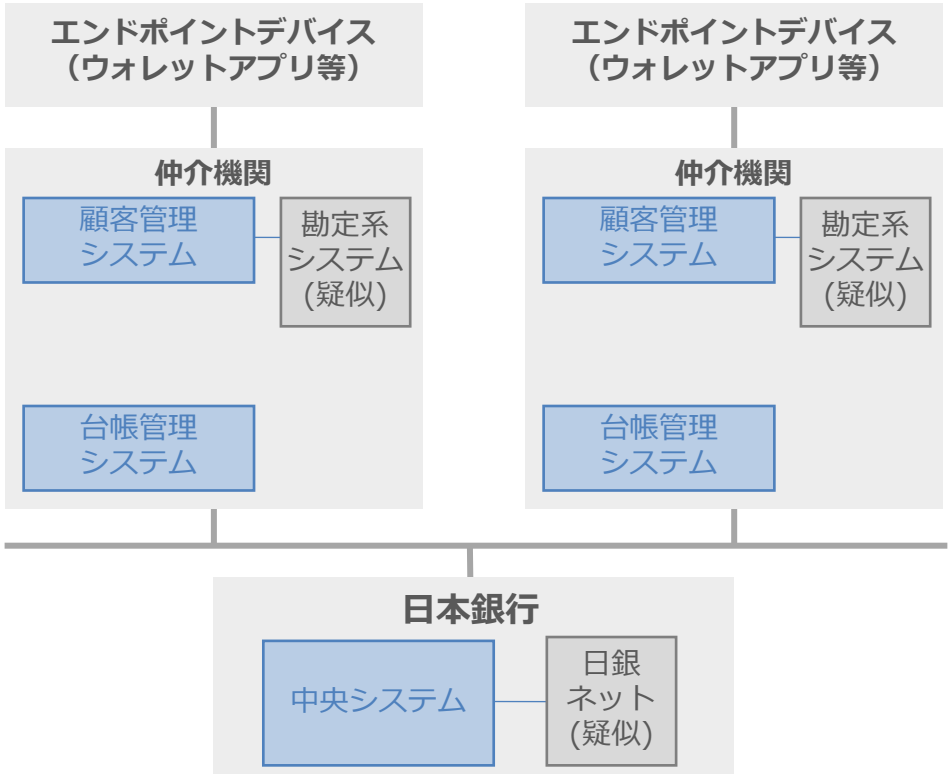
出所：日本銀行 CBDCフォーラムWG4資料

（出所）CBDCフォーラムWG4 第6回会合資料（ソラミツ株式会社様）より抜粋

パイロット実験システムの概要

実験用システムの構成と特徴

システム構成

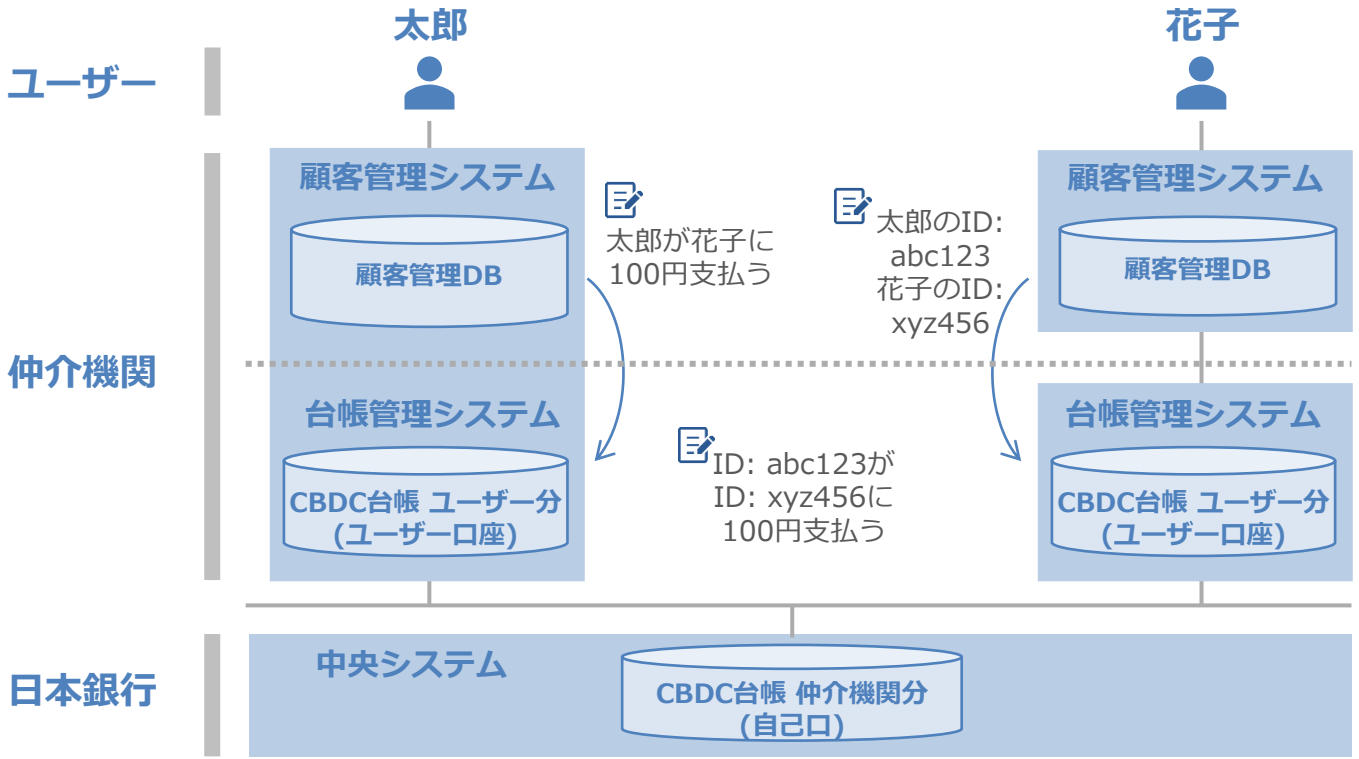


主な特徴

- **①プライバシー配慮**
  - ユーザー情報・取引情報を扱う顧客管理部分と、決済に必要な情報のみを扱う台帳管理部分に分離
- **②送金の処理フロー**
  - 複数の主体が関与することを想定した処理フローを新たに構築
- **③性能への配慮（並列処理向上策）**
  - レコード分割の仕組みを実装することで、並列処理性を高め性能を向上
- **④機能拡張性への配慮**
  - 後から機能を追加しやすくする工夫策を設計段階から埋め込み

実験用システムの構成と特徴：プライバシー配慮

- 仲介機関システムについて、顧客管理システムと台帳管理システムに分離したうえで、顧客管理システムでは、ユーザー情報・取引情報を扱い、台帳管理システムでは、決済に必要な情報のみを扱う仕組みとしている。



※左側（太郎側）は、顧客管理と台帳管理を同一の仲介機関が行う場合を例示

※右側（花子側）は、顧客管理と台帳管理を別々の仲介機関が行う場合を例示

実験用システムの構成と特徴：Tx集中への性能改善アプローチ　#1

- 「レコード分割」の仕組みを実装することで、並列処理性を高め、性能を向上させる工夫を実装（**データ更新時のロック範囲の工夫**）

事例：ユーザーAが100円を持っており、30円出金する場合

従来

ユーザー	残高
A	100　円

30円出金

ユーザー	残高
A	**70** ~~**100**~~　**円**

ロック

出金中は、**Aのレコードはロック**され、その間、Aの取引は不可

今次設計（レコード分割）

ユーザー	残高
A	50　円
	50　円

ユーザー	残高
A	**20** ~~**50**~~　**円**
	50　円

ロック

Aの2行目のレコードはロックされないため、並列的に2行目のレコードで取引可

実装する中での気づき

- ✓ 限界的状況において過検知が発生する可能性（上の事例において、30円の出金中（ロック中）に60円を出金しようとすると、空いているレコードの50円しか使用できず出金できない）

実験用システムの構成と特徴：Tx集中への性能改善アプローチ　#2

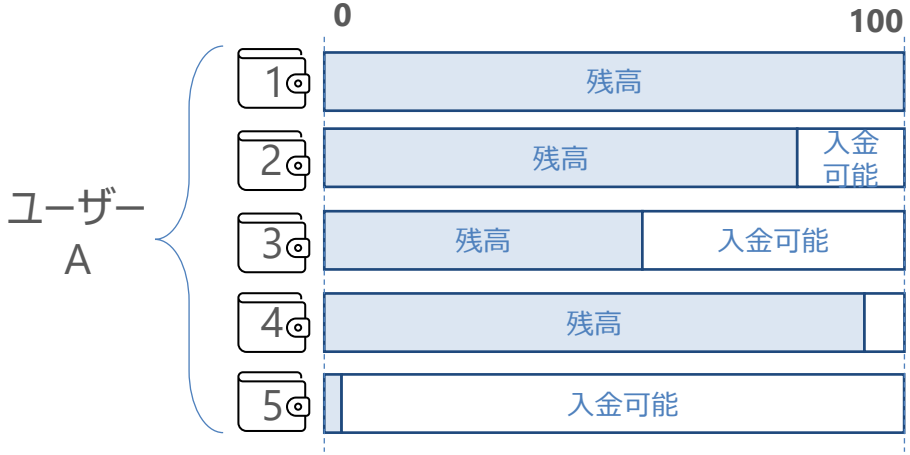
入金時における保有額上限の判定を並列処理するための工夫策

- 赤残チェックに加えて、入金時にチェックする「保有額上限の判定」についても並列処理性を確保する観点から、「保有額上限」を各レコードに分割して管理（これにより各レコードで「入金可能額」を管理できる）
- 入金時には、各レコードに分割された「入金可能額」の部分を使って並列処理を行う。

事例：保有額上限500円、残高レコードを5つに分割する場合

- ✓ 各レコードの保有額上限を100円と設定（100円×5=500円）
- ✓ 100円入金する場合、入金可能額の多いレコードからロックをしてゆき、それらの入金可能額の合計が100円を超えたらロックしたレコードに対して増額処理を開始

残高レコード分割のイメージ



残高レコードのデータベース

ユーザー	保有額上限	残高	入金可能額	
A	**100 円**	100 円	0 円	
	100 円	80 円	20 円	
	100 円	60 円	40 円	**ロック**
	100 円	90 円	10 円	
	100 円	10 円	90 円	**ロック**

（左図のイメージをデータベースのレコードで表現したもの）

実装する中での気づき

- ✓ 限界的状況において過検知が発生する可能性がある点は不変（上の事例では、空いているレコードの入金可能額を足し上げた最大30円しか入金できない）

実験用システムの構成と特徴：Tx集中への性能改善アプローチ　#3

- 顧客管理システムに実験用に設けた累計取引回数・金額をチェックする機能について、履歴を追記して、取引の都度集計する方式で実装（**データ更新の仕方の工夫**）
  - レコード分割方式や、バッチの集計と都度集計とを組み合わせる（例：１日に１回集計を行い、当日分のみ都度集計する）方式も考えられるが、今回は実験のためシンプルかつレコード分割と異なる方式を採用。

事例：ユーザーAが4回目の取引で、30円出金する場合

更新方式

ユーザー	累計取引回数	累計取引額	
A	3　回	500　円	**参照**

取引時には、Aのレコードを参照

30円出金

ユーザー	累計取引回数	累計取引額	
A	**4 ~~3~~　回**	**530 ~~500~~　円**	**ロック**

履歴更新中は、**Aのレコードはロック**され、
その間、Aの取引は不可

今次設計（追記＋都度集計）

ユーザー	取引番号	取引金額	
A	001	200　円	**集計**
A	002	200　円	
A	003	100　円	
A	**004**	**30　円**	**追記**

- 取引時にユーザーAのレコードを**集計**
  （累計取引回数３回、累計取引金額500円）
- 既存のレコードを**ロックしない**ため、並列的に取引可

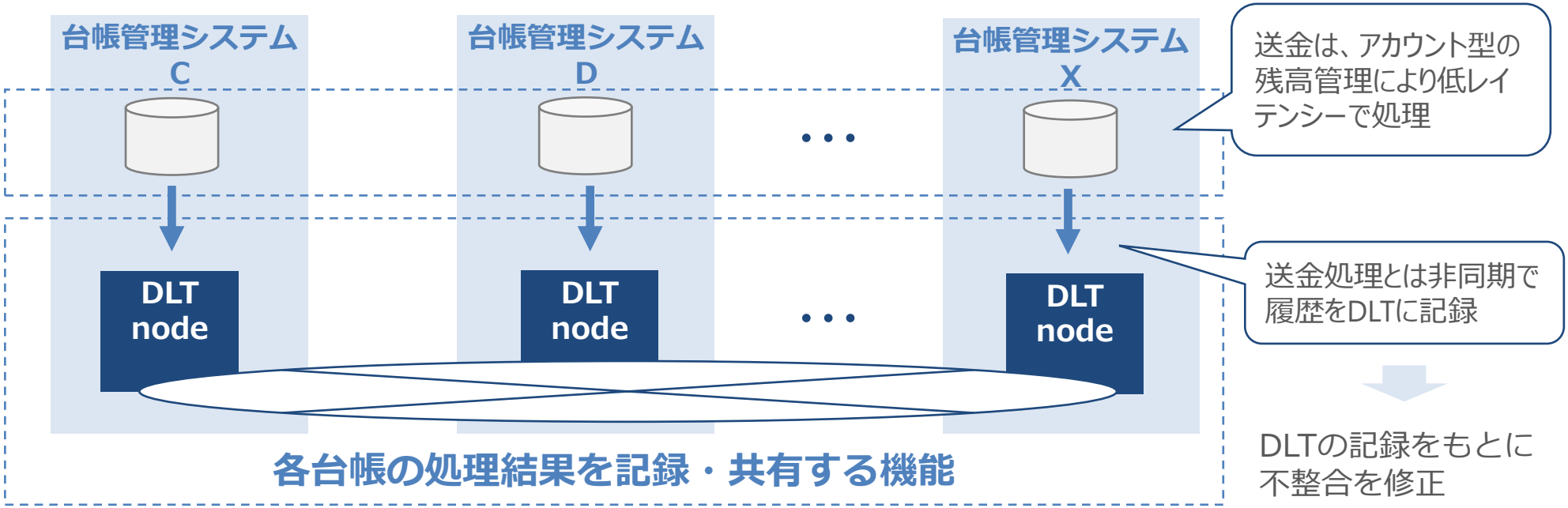
実装する中での気づき

✓ 並列処理する場合における厳密な取引回数・金額のチェックに工夫が必要

机上検討結果：台帳の履歴管理（DLT活用可能性）

- 実験の結果、性能の観点では、口座残高型においてレコード分割により並列処理性が大きく向上することを確認した他、一連の性能試験の間、実装した送金の処理フローにおいて台帳間の不整合は発生しなかった
- もっとも、万が一のデータ消失やデータ改ざんにより台帳間の不整合が生じた場合への備え（レジリエンス）として、各台帳の処理結果を記録・共有する仕組みが有効な可能性

その実現手段の１つとしてWG4で提案いただいたハイブリッドな構成が考えられる



（参考）実験用システムにおけるCBDC台帳の設計パターン

実験用システムは下図の仕組みで構築を実施。**仲介機関を跨ぐ送金であっても、中央システムを経由しない構造**を想定。