自律分散型データ共有・更新システムにおける共有レポジトリの実装について

若林　勇弥† 　沖野　雄哉†† 　清水　敏之†† 　加藤　弘之††† 　吉川　正俊††

† 京都大学工学部情報学科　〒 606–8501 京都府京都市左京区吉田本町
†† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町
††† 国立情報学研究所アーキテクチャ科学研究系　〒 101–8430 東京都千代田区一ツ橋 2-1-2
E-mail: †wakabayashi@db.soc.i.kyoto-u.ac.jp, ††okino@db.soc.i.kyoto-u.ac.jp
††{tshimizu, yoshikawa}@i.kyoto-u.ac.jp †††kato@nii.ac.jp

あらまし　現在では，単一の管理主体がデータを生成・更新して利用する場合よりもむしろ，複数の管理主体がデータを生成・更新したうえで共有し，そのデータを各管理主体において利用する場合のほうが多い．さらに，それらの管理主体（以降，ピアと呼ぶ）それぞれが，どのようなデータを受け入れ，または共有するのかを定めた条件（以降，ポリシーを呼ぶ）をもち，そのポリシーに従ってデータを利用したいという要求は強い．本論文では，中央集権的なデータベースを想定するのではなく，各ピア同士が P2P ネットワークを通じてデータを選択的に共有することで，各々のポリシーに従ってデータの共有を行う自立分散型データ共有・更新システムを設計する．
キーワード　P2P, 双方向変換, ブロックチェーン

1 はじめに

現在，膨大なデータが様々な管理主体によって生成・更新されている．そして，これらのデータを単一の管理主体が独占的に利用する場合よりもむしろ，複数の管理主体があり，各管理主体が自ら生成したデータを他の管理主体に送付したり，逆に他の管理主体が生成したデータを受け取ったり，あるいは受け取ったデータを更新して再度他の管理主体に送付したりするなどしてデータを共同利用する場合のほうが多い．

また，現実の応用を考えた場合，これらのデータの管理主体 (以降, ピア (peer) と呼ぶ) は, 自らのピアで生成または更新したデータをすべて他のピアに送付し, 逆に他のピアから自らのピアに送付されたデータをすべて受け取るというよりもむしろ, 各ピアがデータを送付する条件や受け取る条件 (以降, ポリシーと呼ぶ) をあらかじめ定めておき, そのポリシーに合致するデータに限って送受信を行うことが多いと考えられる．

本論文では，このようなポリシーに従いデータの共有を行う自律分散型データ共有・更新システム [1] における共有リポジトリの設計を行う．まずは清水ら [1] が提案した自律分散型データ共有・更新システムのアーキテクチャを概観し，続いてそのアーキテクチャの実装に関する考察点を提示することで，より有用なデータ共有・更新システムの構築を目指す．

清水ら [1] が提案したアーキテクチャでは，各ピアごとに所持しその各々が必要とするデータを格納する関係 BT(Base Table) と，複数のピアで共有し共同でデータを格納する関係 SR (Shared Repository) の間に，CT(Control Table) と呼ばれるピア間でスキーマが一致している関係を各ピアが所持することで，各ピア間において BT のスキーマが異なる場合においても，ポリシーに従ったデータの共有を行えるシステムを導入している．

一方，本論文では，SR を BT や CT と同様各ピアで所有することで冗長性を高める実装や，その SR についてデータの削除や更新を禁止することで，データの変更履歴を蓄積する実装，さらにブロックチェーンを用いた実装について議論する．

また，このシステムの使用例として，複数の研究者が論文にキーワードを付すという状況を挙げる．この例において，各研究者は，他の研究者とは独立して，任意の論文に 1 件以上のキーワードを付すことができる．この状況下では，例えば経験が浅い研究者が付すキーワードを利用したくないという要求が考えられるが，その場合「その研究者が生成したデータは受け入れない」といったポリシーを設定した上でこのシステムを利用すれば，この要求を満たすことができる．

以降，2 節において [1] で導入されたシステムの基本的な構造と，先に挙げた例におけるシステムの動作例を説明し，3 節で議論点とそれに対する検討案を述べる．さらに，4 節では関連研究を紹介し，最後に 5 節で本研究をまとめる．

2 本システムの基本的な構造

2.1 関係スキーマ

まず [1] で導入されたシステムの基本的な構造を述べる．このシステムは，前節で述べたとおり，各ピアが所持する関係である BT(Base Table), CT(Control Table) と，複数ピアで共有する関係である SR(Shared Repository) からなる．ここで，前節で述べた「論文にキーワードを付す」という例において，各ピアが必要とするデータを格納する関係である BT(Base Table) の関係スキーマは，式 (1) のとおりとなる．ただし，tid

(tuple id) とは，データの組ごとに与えられるグローバルに一意な値であり，これを BT における主キーとする（以降，関係スキーマの表記において下線を付した属性を主キーとする）．

$$\mathrm{BT} = (\underline{\text{tid}}, \text{論文名}, \text{キーワード}) \tag{1}$$

一方，CT (Control Table) の関係スキーマは式 (2) のとおりである．

$$\mathrm{CT} = (\underline{\text{tid}}, \text{論文名}, \text{キーワード}, \mathrm{P}) \tag{2}$$

ここで，属性 P (Provenance) とはデータの来歴を記録するもので，その型は式 (3), (4) の通り再帰的に定義される．

$$\mathrm{P} := \text{pid} \tag{3}$$
$$:= \mathrm{P@pid} \tag{4}$$

ここで，pid (peer id) とは，各ピアごとに与えられるグローバルに一意な値であり，式 (3) は，その P をもつ組がその pid をもつピアによって生成されたことを表し，式 (4) は，さらにその pid をもつピアによって更新されたことを表す．例えば，P=“p1@p2” をもつ組は，pid=“p1” なるピアで生成され，pid=“p2” なるピアで更新されたことを表す．

最後に，SR (Shared Repository) の関係スキーマは式 (5) のとおりである．

$$\mathrm{SR} = (\underline{\text{tid}}, \text{論文名}, \text{キーワード}, \mathrm{P}, \underline{\text{acc}}) \tag{5}$$

ここで，属性 acc (accepting pid) とは，その組を受け入れたピアの pid の集合である．たとえば，acc=“{p1, p2}” を持つ組は，pid=“p1” なるピアと pid=“p2” なるピアで受け入れられていることを表す．SR においては，tid と acc 属性が主キー（複合キー）を構成する．

本システムにおいては，ここまでに述べた P, acc といった属性を記録することで「特定のピアで生成されたデータを受け入れない」「特定のピアで受け入れられたデータを受け入れる」といったポリシーに沿ったデータの送受信を可能にしている．

2.2 ポリシー

各ピアがもちうるポリシーは，次の 2 通りに分類できる．

Export Policy 自らのピアが生成・更新したデータのうち，どのようなデータを公開するのかを定めた条件

Import Policy 他のピアが生成・更新したデータのうち，どのようなデータを受け入れるのかを定めた条件

本システムにおいて，あるピアが他のピアにデータを共有する（以降，Export する）ときは，BT から Export Policy に当てはまるデータのみを CT にコピーし，CT から SR へはすべてのデータをコピーする．一方，あるピアが他のピアからデータを受け入れる（以降，Import する）ときは，SR から Import Policy に当てはまるデータのみを CT にコピーし，CT から BT へはすべてのデータをコピーする．このようにデータを送受信することで，ポリシーに沿ったデータの共有が可能となる．

ここまで述べた関係と，ポリシーに沿った Import, Export の方法を図示すると，図 1 のようになる．

図 1 BT, CT, SR とポリシーに沿った Import, Export に伴うデータの遷移を表した図．点線で示した矢印に沿って各ポリシーに合致するデータのみがコピーされ，実線で示した矢印に沿ってすべてのデータがコピーされる．

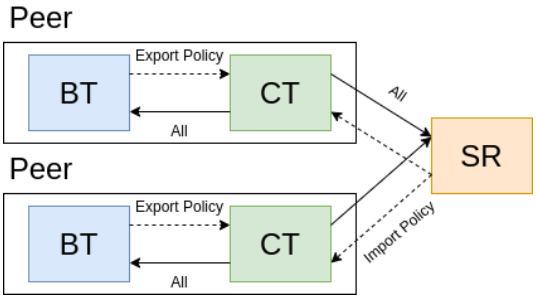


表 1 各ピアの BT

(a) $\mathrm{X_{BT}}$

tid	論文	キーワード
t1	A	データベース
t2	B	情報検索

(b) $\mathrm{Y_{BT}}$

tid	論文	キーワード
t3	A	情報図書館
t4	C	後で見る

(c) $\mathrm{Z_{BT}}$

tid	論文	キーワード
t5	B	最適化

表 2 各ピアのポリシー

ピア	ポリシー	内容
X	Export Policy	すべて Export
X	Import Policy	すべて Import
Y	Export Policy	“後で見る” 以外 Export
Y	Import Policy	Z で生成されたデータのみ Import
Z	Export Policy	すべて Export
Z	Import Policy	一切 Import しない

2.3 例

ここで，先に述べた「各研究者が論文にキーワードを付す」という例を考える．まず，3 人の研究者を仮定し，そのそれぞれを pid=“X”, “Y”, “Z” [1]なるピアに対応させる（以降，それぞれのピア自体のことを単純に X, Y, Z と表記する）．次に，各ピアにおける BT のインスタンス [2]をそれぞれ表 1 のとおりとする．

さらに，各ピアのポリシーを表 2 のとおりとする．

このとき，まずピア X, Y, Z においてデータを Export すると，BT から CT へは各々の Export Policy に合致するデータ

1：先述の通り pid はグローバルに一意な値なので，実運用上は pid として UUID を与えるなどの方法をとるべきである．tid についても同様．
2：それぞれ $\mathrm{X_{BT}}, \mathrm{Y_{BT}}, \mathrm{Z_{BT}}$ と表記する．CT についても同様．

表 3　Export 後のインスタンス

(a) X_{CT}

tid	論文	キーワード	P
t1	A	データベース	X
t2	B	情報検索	X

(b) Y_{CT}

tid	論文	キーワード	P
t3	A	情報図書館	Y

(c) Z_{CT}

tid	論文	キーワード	P
t5	B	最適化	Z

(d) SR

tid	論文	キーワード	P	acc
t1	A	データベース	X	{X}
t2	B	情報検索	X	{X}
t3	A	情報図書館	Y	{Y}
t5	B	最適化	Z	{Z}

表 4　Import 後のインスタンス

(a) SR

tid	論文	キーワード	P	acc
t1	A	データベース	X	{X}
t2	B	情報検索	X	{X}
t3	A	情報図書館	Y	{X, Y}
t5	B	最適化	Z	{X, Y, Z}

(b) X_{CT}

tid	論文	キーワード	P
t1	A	データベース	X
t2	B	情報検索	X
t3	A	情報図書館	Y
t5	B	最適化	Z

(c) X_{BT}

tid	論文	キーワード
t1	A	データベース
t2	B	情報検索
t3	A	情報図書館
t5	B	最適化

(d) Y_{CT}

tid	論文	キーワード	P
t3	A	情報図書館	Y
t5	B	最適化	Z

(e) Y_{BT}

tid	論文	キーワード
t3	A	情報図書館
t4	C	後で見る
t5	B	最適化

のみがコピーされ，CT から SR へはすべてのデータがコピーされるので，各ピアにおける CT と，これらのピアで共有する SR のインスタンスは表 3 のとおりとなる．Y の Export Policy に合致しない tid=“t4” なる組は Y_{CT} にコピーされないため，SR にも含まれない．

次に，各ピアがデータを Import するときは， SR から CT へは各々の Import Policy に合致するデータのみがコピーされ，CT から BT へはすべてのデータがコピーされるので，SR と各ピアにおける CT と BT のインスタンスは表 4 の通りとなる．あるピアがある組のデータを受け入れると，SR に含まれるその組の acc 属性に，そのピアの pid が追加される．

さらにここで，Y において tid=“t5” なる組のキーワードを“最尤推定” に更新すると，BT, CT, SR のインスタンスは表 5 の通りとなる．この更新によって，(tid, 論文, キーワード) = (“t5”, “B”, “最適化”) なる組は Y で受け入れられたデータではなくなったため，SR においてこの組に対応する acc 属性から Y が削除され，新たに (tid, 論文, キーワード, P, acc) = (“t5”, “B”, “最適化”, “Z@Y”, “{Y}”) なる組が SR に追加される．

表 5　更新後のインスタンス

(a) Y_{BT}

tid	論文	キーワード
t3	A	情報図書館
t4	C	後で見る
t5	B	最尤推定

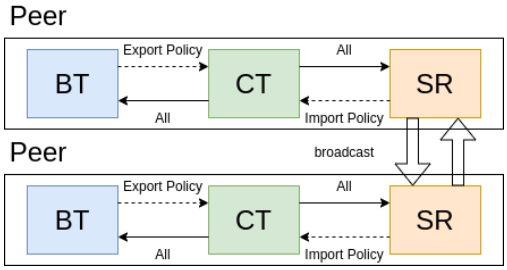
(b) Y_{CT}

tid	論文	キーワード	P
t3	A	情報図書館	Y
t5	B	最尤推定	Z@Y

(c) SR

tid	論文	キーワード	P	acc
t1	A	データベース	X	{X}
t2	B	情報検索	X	{X}
t3	A	情報図書館	Y	{X, Y}
t5	B	最適化	Z	{X, Z}
t5	B	最尤推定	Z@Y	{Y}

図 2　SR を分散化したシステム



3　議論点と検討案

3.1　中央集権的な SR の分散

前節で導入したシステムにおいて，各ピアは他のピアの影響を受けずに独立してポリシーを設定し，データの生成や更新，また Import や Export を行うことができるため，基本的には自律分散的なシステムであると評価することができる．一方，SR については複数のピアで共有しているので，中央集権的であると評価することができる．

SR が中央集権的であることによる弊害として，SR に障害が起こった際にシステム全体が致命的な影響を受けるという点が挙げられる．そこで，図 2 のように SR を BT, CT と同様各ピアにおいて所持し，各ピアにおいて SR が更新されたらその更新内容を他のピアの SR に一斉送信するというシステムを提案し，ここで導入する SR を「分散型 SR」とよぶ．

3.2　追記型 SR の導入

前節において述べたとおり，Y が (tid, 論文, キーワード) = (“t5”, “B”, “最適化”) なる組を更新し (“t5”, “B”, “最尤推定”) とすると，SR において (tid, 論文, キーワード, P, acc) = (“t5”, “B”, “最適化”, “Z”, “{X, Y, Z}”) なる組が (“t5”, “B”, “最適化”, “Z”, “{X, Z}”) に更新される．そのため「Y がかつて (tid, 論文, キーワード) = (“t5”, “B”, “最適化”) なる組を保持していた」という履歴情報をこのシステムから取り出すことは不可能である．この場合，例えば「かつて Y が Import したことのあるデータを Import する」という Import Policy を実現することができない．

表 6　更新前のインスタンス
(a) Y_{BT}

tid	論文	キーワード	uid
...	...	...	...
t5	B	最適化	u5

(b) Y_{CT}

tid	論文	キーワード	P	uid
...	...	...	...	...
t5	B	最適化	Z	u5

(c) SR

tid	論文	キーワード	P	acc	uid
...	...	...	...	...	...
t5	B	最適化	Z	+X	u5
t5	B	最適化	Z	+Y	u5
t5	B	最適化	Z	+Z	u5

そこで，SR を，組の削除と更新を許さず，組を末尾に追加することのみを許す順序付き関係とし，その関係スキーマを式 (6) の通りとする．また，この SR を「追記型 SR」とよぶ．

$$\mathrm{SR} = (\mathrm{tid}, \text{論文名}, \text{キーワード}, \mathrm{P}, \underline{\mathrm{acc}}, \underline{\mathrm{uid}}) \tag{6}$$

従来のシステムとの相違点として，次の点が挙げられる．

- 前節のように，既に SR に格納された組の acc 属性を更新することはできない．そのため，acc は pid の集合ではなく，acc 属性にある pid を追加したい場合は acc = “+pid” なる組を，逆に acc 属性からある pid を削除したい場合は acc = “–pid” なる組を SR の末尾に追加する．
- すべての更新履歴を残すため，{tid, acc} を主キーにするのではなく，新たに uid (unique id) を導入して {uid, acc} を主キーとする．この uid は，各ピアの BT においてデータを更新するたびに新たに付与される．これに伴い，各ピアの BT, CT のスキーマにも属性 uid を追加し，これを主キーとする．

追記型 SR を導入したシステムにおいて，Y が tid=“t5” なる組のキーワードを “最適化” から “最尤推定” に更新すると，BT, CT, SR のインスタンスは表 6 の状態から表 7 の状態に遷移する．

この場合，追記型 SR から「かつて Y が Import したことのあるデータ」を取り出したいときは acc = “+Y” なる組をすべて取り出せばよく，また「Y が現在 Import しているデータ」を取り出したい場合は，それぞれの tid をもつ組について「acc = “+Y” または acc = “–Y” なる組のうち，最も末尾に位置しているデータ」を取り出せばよい．ただし，ある tid を持つ組のうち，acc = “–Y” なるデータが最も末尾に位置していた場合は「Y は現在その組の tid を持つデータを Import していない」とみなす．

3.3　各ピアの SR における組の順序不一致問題

3.1 小節, 3.2 小節で提案した SR を導入すると，SR の順序がピア間で異なる状況が発生することがある．例えば，pid = “P”, “Q” なるピア（以降，それぞれのピア自体のことを単純に P, Q と表記する）がほぼ同時にデータを Export すると，それぞれのピアにおける SR [3]のインスタンスが表 8 から表 9 の状態に遷移するため，ピア間で順序が異なる SR のインスタンスをもつことになる．

表 7　更新後のインスタンス
(a) Y_{BT}

tid	論文	キーワード	uid
...	...	...	...
t5	B	最尤推定	u5-1

(b) Y_{CT}

tid	論文	キーワード	P	uid
...	...	...	...	...
t5	B	最尤推定	Z@Y	u5-1

(c) SR

tid	論文	キーワード	P	acc	uid
...	...	...	...	...	...
t5	B	最適化	Z	+X	u5
t5	B	最適化	Z	+Y	u5
t5	B	最適化	Z	+Z	u5
t5	B	最適化	Z	–Y	u5
t5	B	最尤推定	Z	+Y	u5-1

表 8　一斉送信前のインスタンス
(a) P_{SR}

tid	論文	キーワード	P	acc	uid
t6	D	トランザクション	P	+P	u6

(b) Q_{SR}

tid	論文	キーワード	P	acc	uid
t7	E	ネットワーク	Q	+Q	u7

表 9　一斉送信後のインスタンス
(a) P_{SR}

tid	論文	キーワード	P	acc	uid
t6	D	トランザクション	P	+P	u6
t7	E	ネットワーク	Q	+Q	u7

(b) Q_{SR}

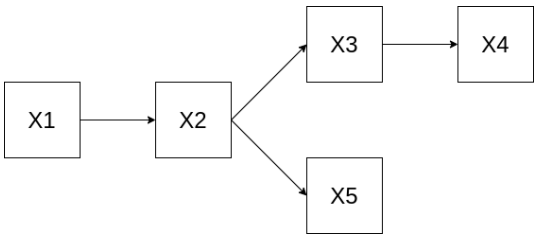
tid	論文	キーワード	P	acc	uid
t7	E	ネットワーク	Q	+Q	u7
t6	D	トランザクション	P	+P	u6

3.2 小節において SR は順序付き関係であると定義したため，P_{SR}, Q_{SR} はそれぞれ異なるインスタンスである．先述したとおり 追記型 SR は「現在ある pid を持つピアにおいて Import されている組のうち，ある tid を持つ組」を取り出す際は「acc = “+pid” または acc = “–pid” なる組のうち，最も末尾に位置している組で，かつその tid を持つ組」を取り出す必要があるため，順序が異なるとこの結果が異なる場合がある．

表 9 の場合はそれぞれのピアにおいて Export した組の tid

3 : それぞれを P_{SR}, Q_{SR} と表記する．

図 3　tid = “t1” なる組を複数のピアが更新した順番を示す図



が異なるため問題は発生しないが，より複雑な更新が行われた場合における組の順序不一致問題については議論が必要である．

例えば，pid = “X1”, “X2”, ..., “X5” なる 5 つのピアが存在し，そのそれぞれを単純にピア X1, X2, ..., X5 と呼ぶことにする．ここでピア X1 が tid = “t1” なる組を生成しこれを Export した後，その組をピア X2, X3, X4 の順で更新したとする．また，それと並行してピア X2 が更新した後にピア X5 が更新したとする．つまり，この更新順を表した図 3 の通り，単一の組についてその更新履歴が枝分かれしている状況が発生したとする．

ここで，ピア X4 における更新と X5 における更新が同時に行われたとき，各ピアの SR において，tid = “t1“ なる組のうち，P = “X1@X2@X3@X4” なる組と P = “X1@X2@X5” なる組のどちらが下に格納されるかが異なる場合があり，この場合は，例えば「すべての組について，最新の組を Import する」というポリシーを持つピアが Import する tid = “t1” なる組が，一意に定まらないことになる．

このように，単一の組について複雑に更新が行われた場合は，各ピアにおける SR 間で組の順序が一致しないことがあり，この場合同じポリシーを持つピアであっても Import/Export を行うべき組に関する判定結果が異なる場合が存在する．

4　関 連 研 究

4.1　双方向変換

本論文では図 1, 2 の通り， BT–CT 間及び CT – SR 間のデータの送受信を「すべてコピー」または「ポリシーに当てはまる組のみコピー」のいずれかで指定したが，ポリシーに沿ったデータのやり取りを厳密に定義する方法として図 4 のように双方向変換 [3] を用いるというものがある．

双方向変換とは，図 5 のように関係を引数としてビューを求める変換操作を get とし，関係とある更新が行われたビューを引数としてその更新が行われた関係を求める変換操作を put としたときの変換の組 {get, put} を求めること，またはその変換の組自体のことを指す [3]．

また，この get, put を datalog を用いて定義する試みが [4] でなされている．例えば，2 節において Y の Export Policy を「“後で見る” 以外 Export」としたが，これに相当する get を datalog で記述すると式 (7) の通りとなる．これに対して put に相当する datalog を求めることで双方向変換を定義する．

図 4　双方向変換を用いて BT–CT, CT–SR 間のデータのやり取りを行うアーキテクチャ

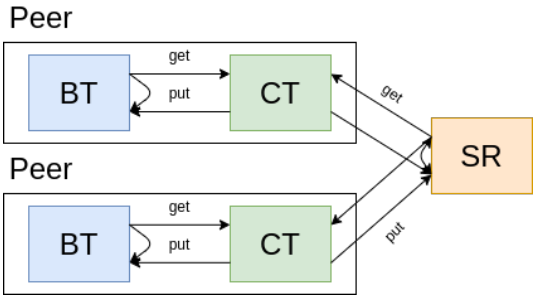
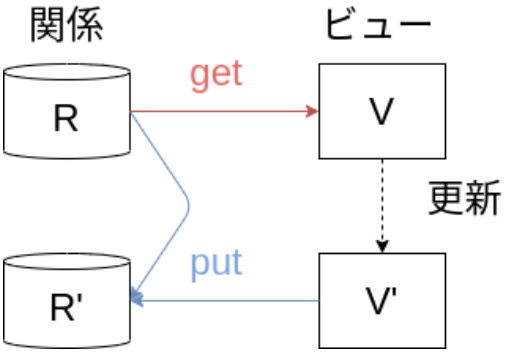


図 5　双方向変換



$$Y_{CT}(\underline{\text{tid}}, \text{論文}, \text{キーワード}, \text{“Y”}) := Y_{BT}(\underline{\text{tid}}, \text{論文}, \text{キーワード}), \text{キーワード} \neq \text{“後で見る”} \quad (7)$$

各ピアにおける BT のスキーマが異なる場合でも，CT のスキーマを各ピア間で統一し，BT のインスタンスを引数として CT のインスタンスを求める get を含む双方向変換を各ピアについて求めることで，ポリシーに沿ったデータの送受信が可能になると考えられる．

4.2　Dejima アーキテクチャ

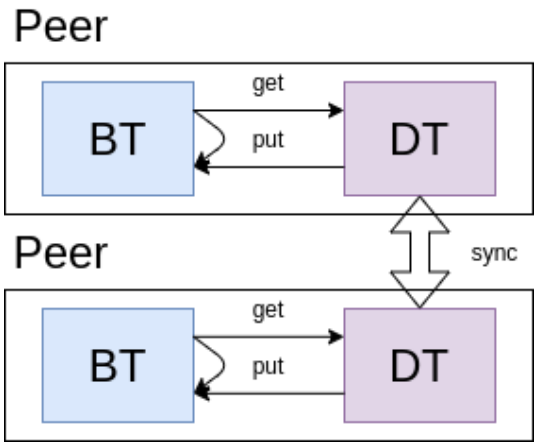
4.1 小節で言及した双方向変換を用いたデータ共有・更新システムの一つとして [5] で Dejima アークテクチャというシステムが導入されている．Dejima アーキテクチャとは，図 6 のように各ピアが保有する情報を格納する BT (Base Table) と他のピアに共有したい情報を格納する DT (Dejima Table) を双方向変換でつなぎ，各ピアの DT 同士のデータを単純に同期させるシステムである．このシステムを用いることで，本論文で導入したシステムと同様 Export Policy に沿ったデータの Export が可能となる．

ただし，自分のピアの DT と同期する他のピアの DT に格納されたデータは，DT 同士の同期によって必ず自分のピアにも格納されることになるため，Import Policy を Export Policy と独立して定義することはできないといえる．

4.3　ブロックチェーン

3 節で「分散型 SR」「追記型 SR」を導入したが，これらの性質を併せ持った SR の性質は，ブロックチェーンが持つ性質と相似している．なぜならば，ブロックチェーンは中央集権的

図 6　Dejima アーキテクチャ



なリポジトリを仮定せず，各ピア同士で P2P ネットワークを形成し情報を一斉送信することで情報を共有する仕組みであり，かつ一度ピア同士で合意した情報を改ざんできる可能性が時間経過に伴い漸減するという特徴を持つからである [6].

そのため，3 節で導入した分散型かつ追記型の SR をブロックチェーンで実装することができると考えられる．

さて，ブロックチェーンを用いた仮想通貨技術である bitcoin においては，UTXO（未使用残高; Unspent Transaction Output） と呼ばれる技術が用いられている．この技術について概観し，本アーキテクチャへの応用可能性を議論する．

まず，bitcoin を用いて取引を行うピアは，取引を行うごとにアドレスを発行し，自らのピアとそのアドレスの関連を記録しておく．ただし，この関連は他のピアに開示しない．そして，トランザクションと呼ばれる入力と出力の組をブロックチェーンに記録することで取引を行う．このうち，入力は「通貨量」と「送金元のアドレス」の組で構成され，出力は「通貨量」と「送金先のアドレス」の組で構成される．また，トランザクションのうち出力が記録されていないものを UTXO と呼び，これが送金元のアドレスに対応するピアの「残高」となる．

例えば，表 10(a) は UTXO の例であり，アドレス α に関連付けられた 10 BTC の UTXO を表す．

アドレス α に対応するピアが他のピアに支払いを行う場合は，出力として，送金したい通貨量と送金先のアドレスを記録する．ただし，入力に記録する通貨量の和と，出力におけるそれは一致する必要がある．そのため，例えばアドレス α からアドレス β に 8BTC を支払いたい場合は，表 10(b) の通り 8BTC をアドレス β に関連づけ，2BTC をアドレス α に関連付ける．また，これと同時に，新たな UTXO として，表 10(c), (d) が生成される．

つまり，bitcoin においては，UTXO を使用済みトランザクションに変換することで支払い記録を残し，その結果新たな UTXO を作成することで，新たな支払いに利用する．

一方，UTXO は仮想通貨技術と不可分な技術ではなく，同じくブロックチェーンを用いた仮想通貨技術である Ethereum においては，UTXO を用いずに支払いの管理を行っている．

具体的には，支払いの履歴と各ピアにおける残高をそれぞれ管理し，支払いの履歴だけをブロックチェーンで記録するという仕組みを用いている．

表 10　bitcoin におけるトランザクション

(a) UTXO

入力	出力
10BTC	
α	

(b) 使用済み

入力	出力
10BTC	8BTC
α	β
	2BTC
	α

(c) 新規 UTXO(1)

入力	出力
8BTC	
β	

(d) 新規 UTXO(2)

入力	出力
2BTC	
α	

ここで，UTXO を用いる形式と用いない形式を比較する．UTXO を用いると，自分以外のピアについて，どのアドレスがどのピアに対応するのかを知ることができないため，他のピアの残高を知ることが不可能となる．そのため，この技術はプライバシーを保護するという観点において有用であると評価できる．しかし，取引を行うたびに UTXO が発生することになるため，その管理は単純に残高を記録するより複雑となる．

一方，UTXO を用いない場合は，支払い履歴と各ピアにおける残高をそれぞれ管理しているため，各ピアにおける残高を取得するのは容易であると評価できる．

ここで，本論文で取り上げた自律分散型データ更新・共有システムにおいて，各ピアの Export Policy に当てはまる（つまり，SR に Export される）組は，各ピア間において秘匿する必要がない組である．そのため，SR をブロックチェーンで実装する場合，UTXO を用いることでプライバシーを重視するメリットは小さく，むしろ管理が煩雑になるだけである．

そのため，SR をブロックチェーンで実装する場合は，UTXO を用いない形式で実装すべきであると結論付けられる．

5　おわりに

本論文では，各ピアがもつ Export Policy 及び Import Policy に従ってデータを送受信できる自律分散型データ共有・更新システムを設計した．

ここで，Export Policy とは，各ピアがどのようなデータを共有するかを定めたルールであり，Import Policy とは，各ピアがどのようなデータを受け入れるかを定めたルールである．また，そのルールを記述するために，データの来歴を表す P (Provenance) 属性や，データを受け入れているピアを記録する acc (accepting pid) 属性を利用することができる．

このシステムには，各ピアが必要とするデータを格納する BT(Base Table) と，各ピアが Export したデータや，各ピアがどのデータを Import したのかを共有する SR (Shared Repository)，さらに CT(Control Table) と呼ばれる中間関係が更に必要となる．

そして，ピアからデータを Export する際，BT から CT へ

は Export Policy に当てはまるデータのみを反映し，CT から SR へはすべてのデータをコピーする．一方，各ピアがデータを Import する際，SR から CT へは Import Policy に当てはまるデータのみをコピーし，CT から BT へはすべてのデータを反映させる．

ここで，SR の実装について，単一の SR をすべてのピアで共有するか，各ピアが SR を保有した上で，データが更新されたらその内容を一斉送信するという運用にするかといった 2 通りが考えられる．このうち，前者を採用すると，すべてのピアが単一の SR に依存する中央集権的なシステムを構成することになり，この SR に障害が発生した場合にシステム全体が致命的な影響を受けることになる．一方，後者を採用すると自律分散的なシステムを構成することになり，冗長性が向上する．これを分散型 SR とよんだ．

また，Export/Import Policy に過去の時点におけるデータを用いるルールを適用したい場合は，SR に過去の時点におけるデータを蓄積する必要がある．そこで，追記型 SR と呼ばれるデータの削除・更新を認めず末尾への追加のみを認める順序付き関係を導入した．

分散型かつ追記型の SR を採用すると，複雑な更新が行われた際に各ピアにおける SR 間で組の順序が一致しなくなる可能性がある．追記型 SR は順序付き関係なので，順序が異なる SR のインスタンス同士は異なるインスタンスであると評価できる．本論文では，ピアのポリシーによっては，同じポリシーであってもそれに当てはまる組の集合が一意に定まらない可能性がある点を指摘した．

さらに，SR をこのような形式にする場合は，ブロックチェーンによる実装が有効である．この際，UTXO と呼ばれる，仮想通貨技術において未使用残高を管理する仕組みを採用するかどうかを議論したが，本アーキテクチャにおいてブロックチェーンを利用する場合は，パフォーマンスの観点から UTXO を用いない運用を行うほうが適切だと判定した．

最後に，関連研究として，双方向変換と Dejima Architecture について述べた．

このうち，双方向変換とは，関係からビューを求める操作である get と，更新されたビューを更新される前の関係から更新後の関係を求める操作である put の組ないしはその組を求めることであり，この議論を用いることで，本システムにおける BT–CT 間，CT–SR 間のデータのやり取りを厳密に記述することができる可能性を示唆した．

また，本システムと類似したシステムとして Dejima Architecture を取り上げた．Dejima Architecture は BT(Base Table) と DT(Dejima Table) を用いて各ピアのポリシーに沿ったデータのやり取りを行うシステムであるが，本論文で提案したシステムとは異なり，Import Policy を Export Policy から独立して定義することができない点を指摘した．

以上，各ピアがもつポリシーに従ってデータを自律分散的に送受信できるデータ共有・更新システムを定義し，そのシステムで用いられる SR の実装方法を中心に議論を進めた．

謝辞　本研究の一部は JSPS 科研費 JP17H06099,JP18H04093 の助成を受けたものです．

文　献

[1] 清水 敏之, 加藤 弘之, 吉川 正俊. 『科学メタデータクリーニングのための自律分散型データ共有・更新システムに向けて』, DEIM Forum 2020, D8-2.

[2] Grigoris Karvounarakis, Todd J. Green, Zachary G. Ives, and Va l Tannen. Collaborative data sharing via update exchange and provenance. ACM Transactions on Database Systems, 38(3):19:1–19:42, 2013.

[3] Faris Abou-Saleh, James Cheney, Jeremy Gibbons, James McKinna, and Perdita Stevens. Introduction to bidirectional transformations. 9715:1–28, 2018.

[4] Van-Dang Tran, Hiroyuki Kato, and Zhenjiang Hu. Programmable view update strategies on relations. PVLDB, 13(5):726–739, 2020.

[5] Y. Asano, S. Hidaka, Z. Hu, Y. Ishihara, H. Kato, H. Ko, K. Nakano, M. Onizuka, Y. Sasaki, T. Shimizu, V. Tran, K. Tsushima, and M. Yoshikawa, “Making view update strategies programmable - toward controlling and sharing distributed data -,” CoRR, vol. abs/1809.10357, 2018.

[6] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, ”http://bitcoin.org/bitcoin.pdf.