問合せの代数の実装

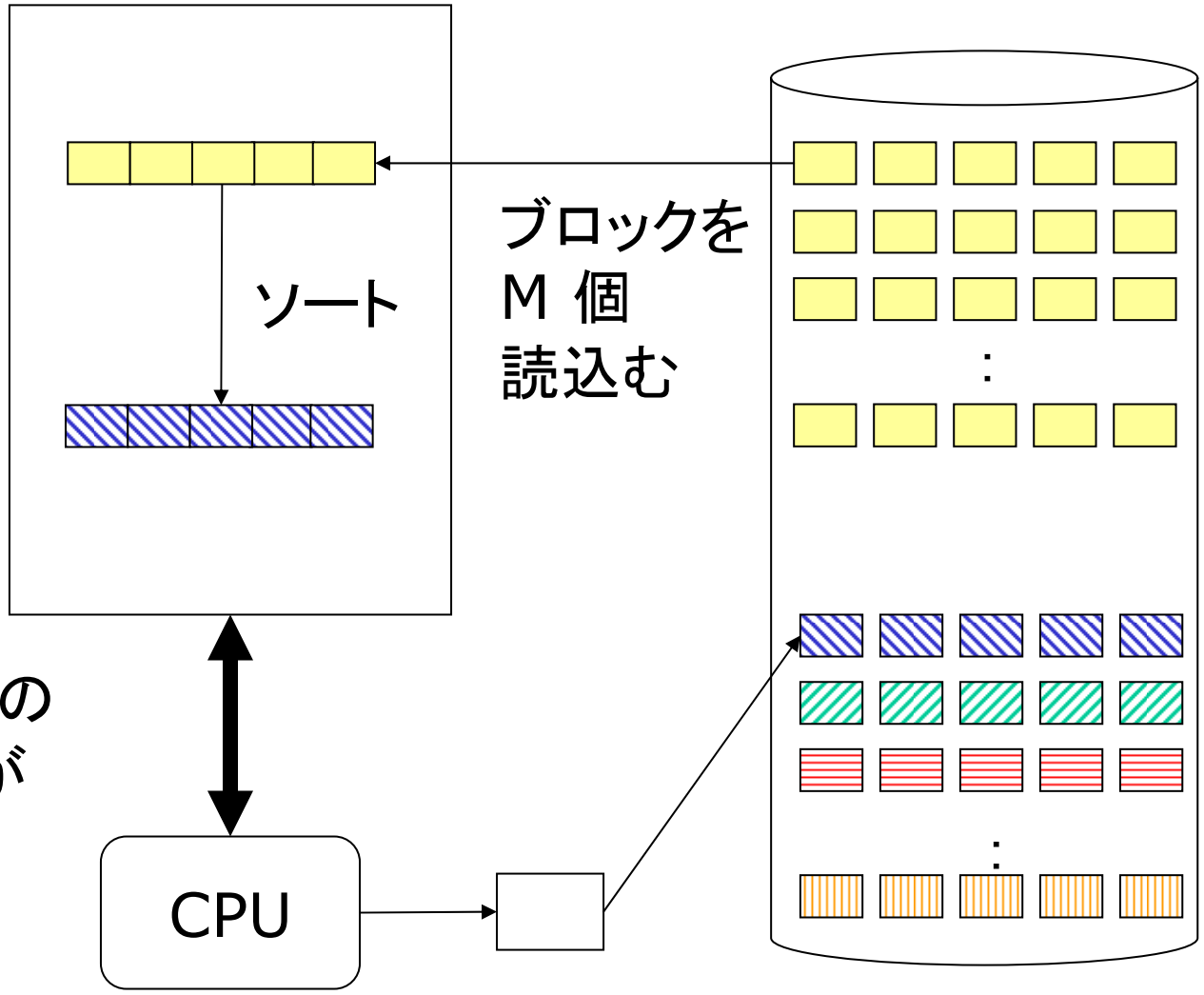
DBを2回走査することで
より大きな DB を処理できる方法

ソートをつかう場合

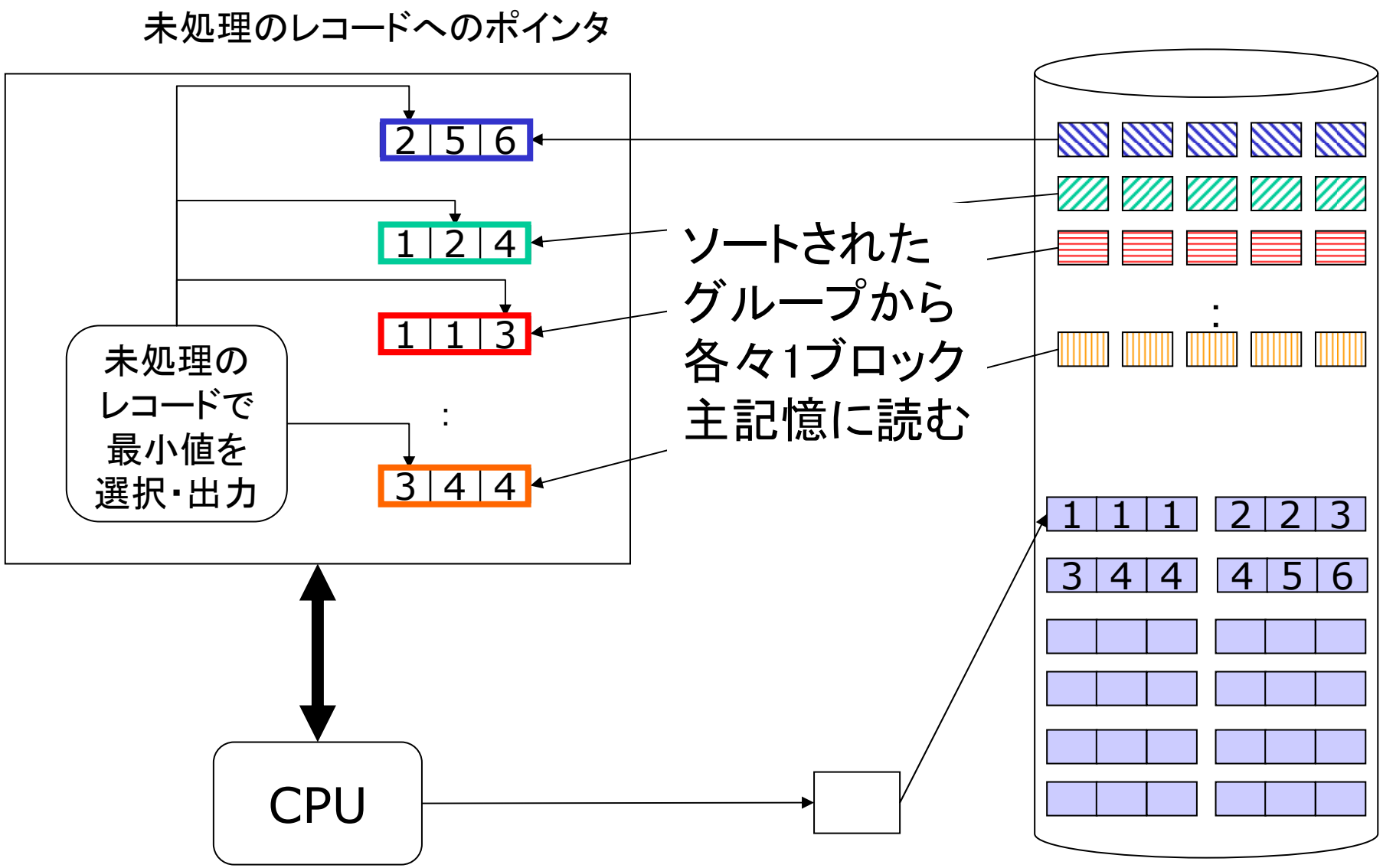
DB を2回走査する方法　ソート

主記憶に入らないDB のソート
Two-Phase Multiway Merge Sort

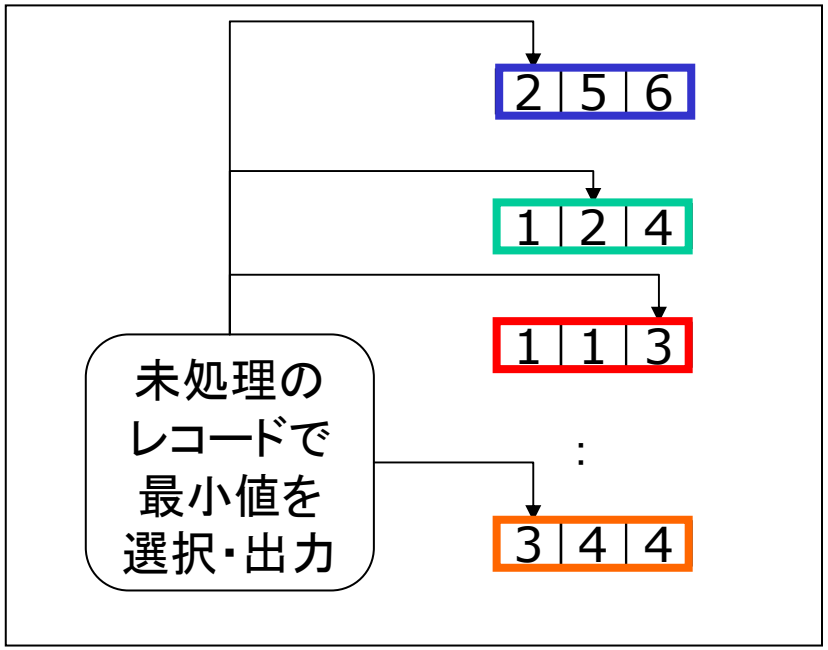
- ブロックを M 個読込む
- 主記憶内でソート
  ブロックを読込む時間内に通常はソートできる
- ソートされた M 個のブロックグループが複数できる

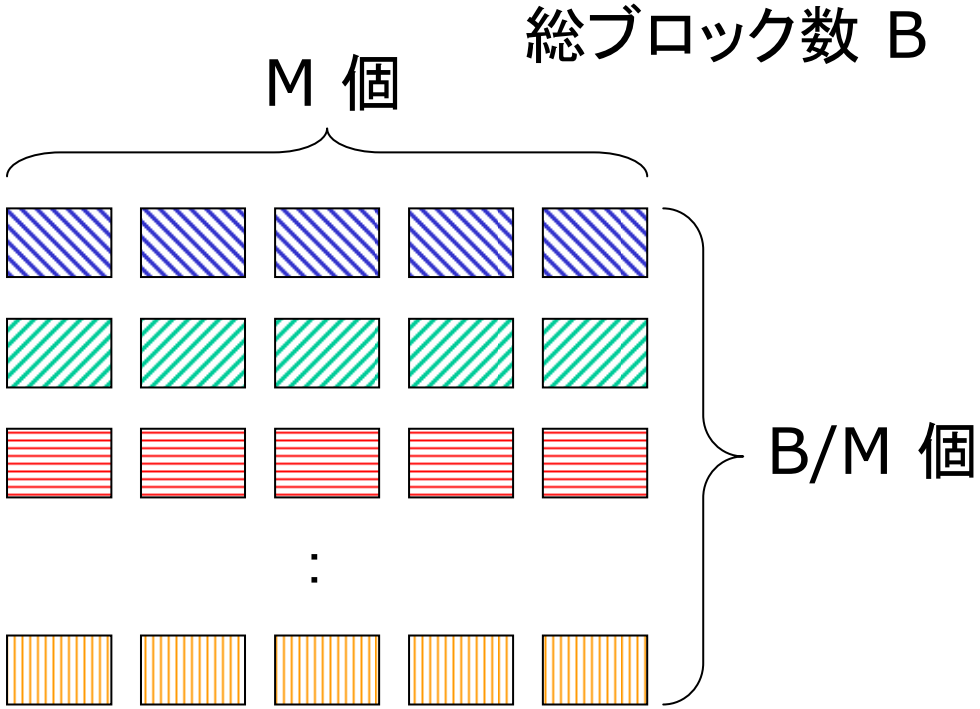


Two-Phase Multiway Merge Sort



Two-Phase Multiway Merge Sort のデータ量の限界



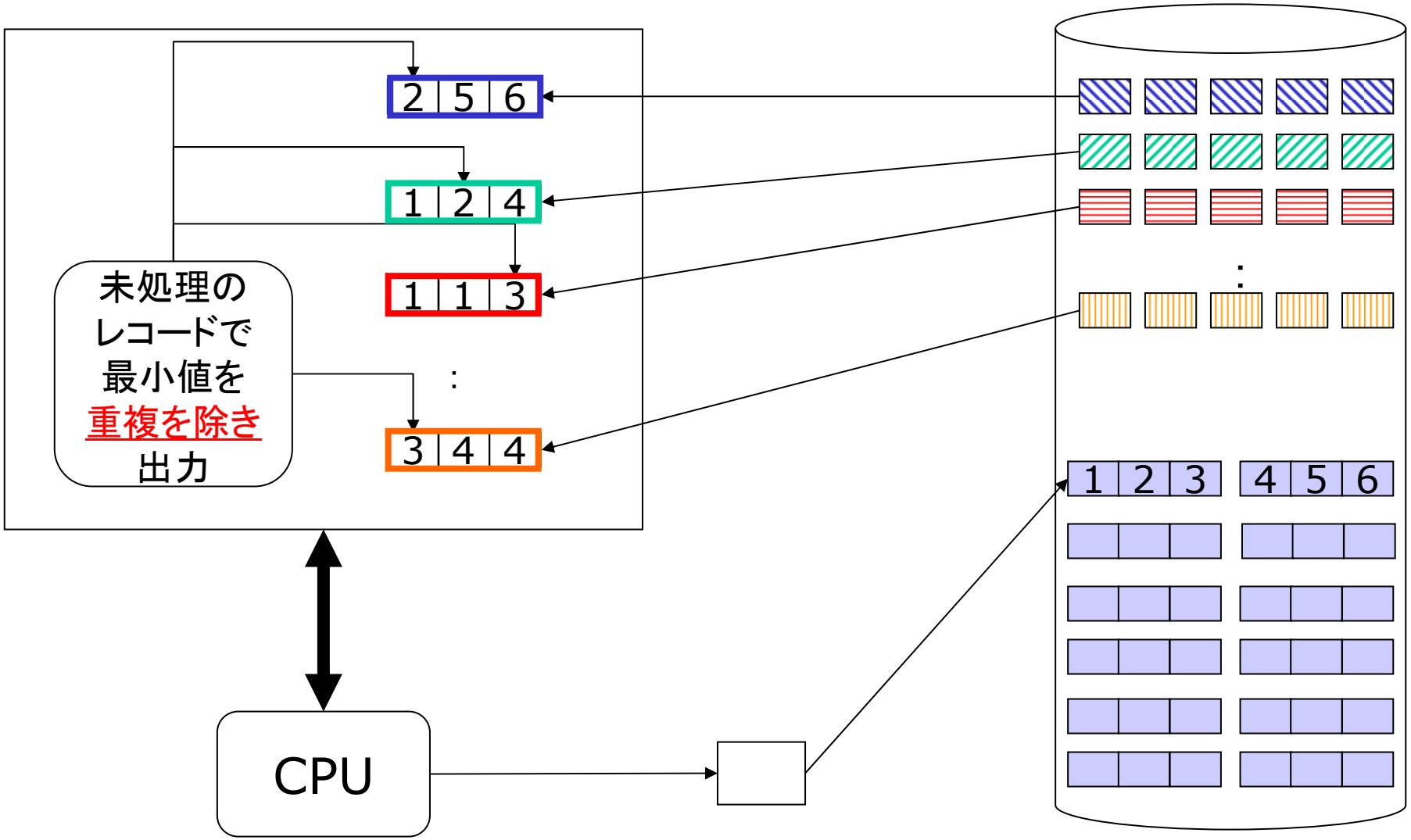


主記憶にブロックを M 個保持できる場合、B/M ≦ M
B ≦ M^2 のデータまでソートできる。　入出力は 4B ブロック。

例　4kBのブロックを, 15msec で転送できる場合の合計時間
M = 10^3, B = 10^6　主記憶4MBで 4GB　を　60,000秒
M = 10^4, B = 10^8　主記憶40MBで 400GB　を　6,000,000秒（約70日）

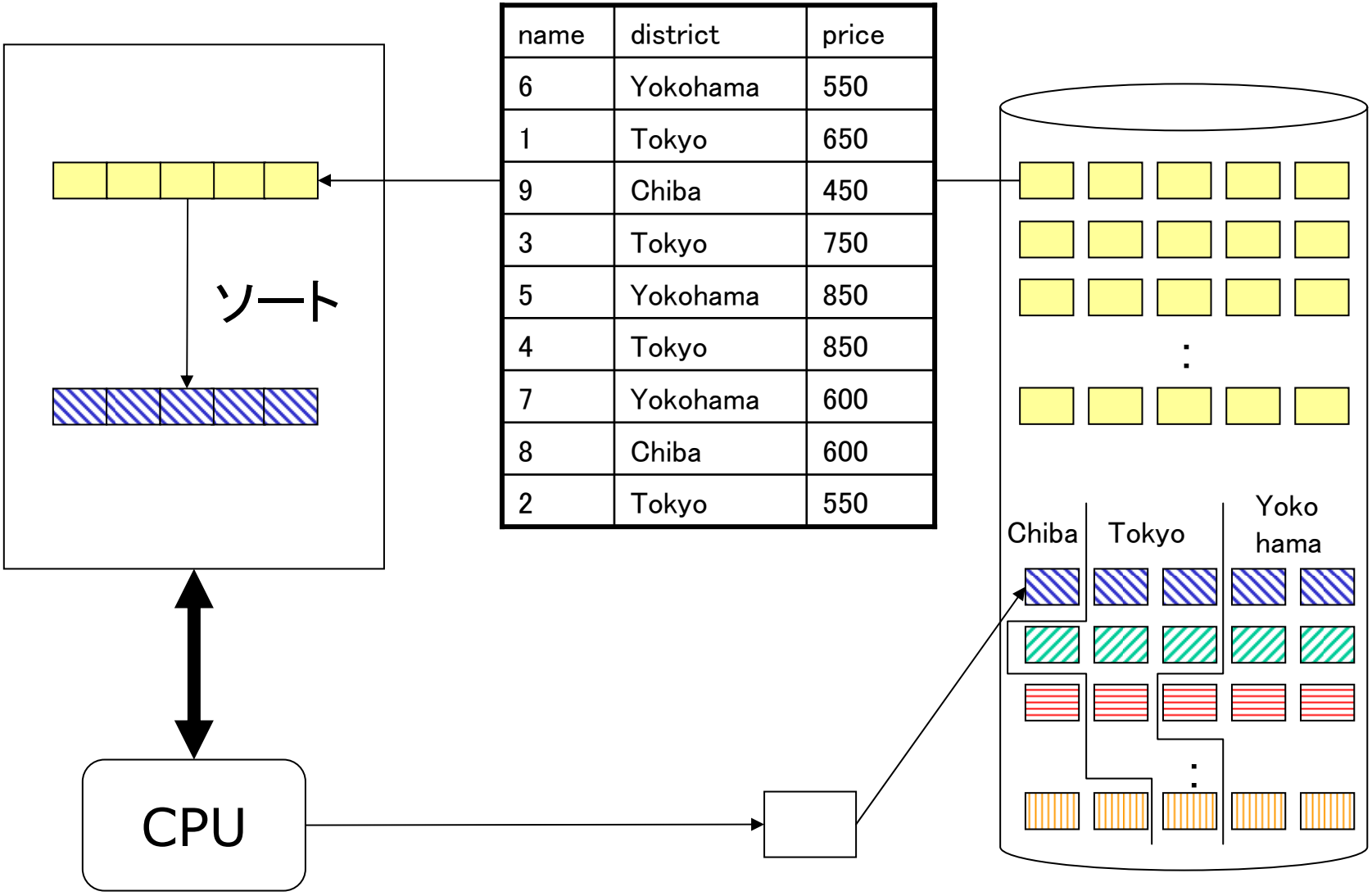
ソートをつかい DB を2回走査して重複除去

M 個のブロック毎にソートしておく



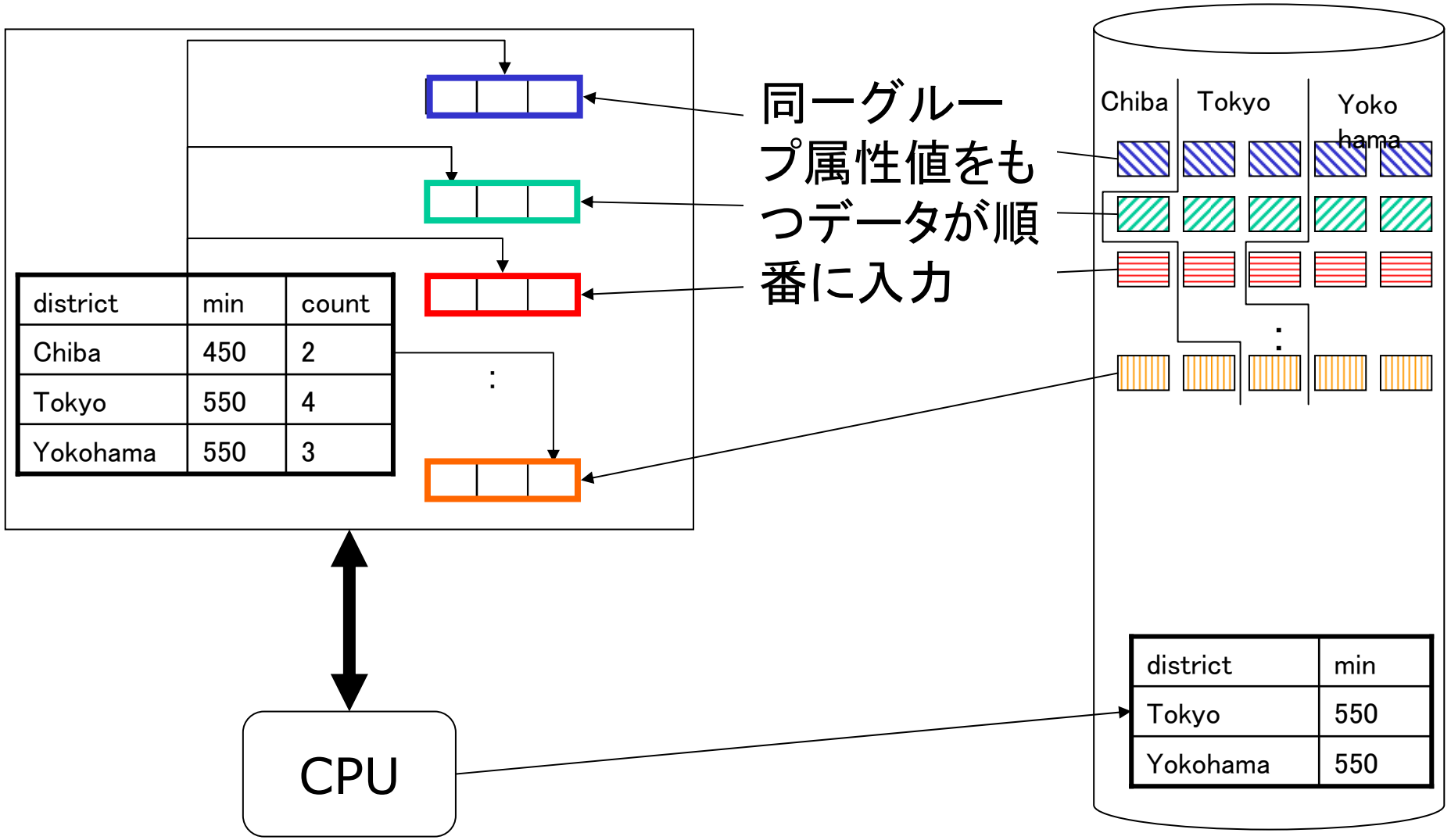
ソートを使ったグループ分けと集約

M 個のブロック毎にグループの対象属性でソートしておく



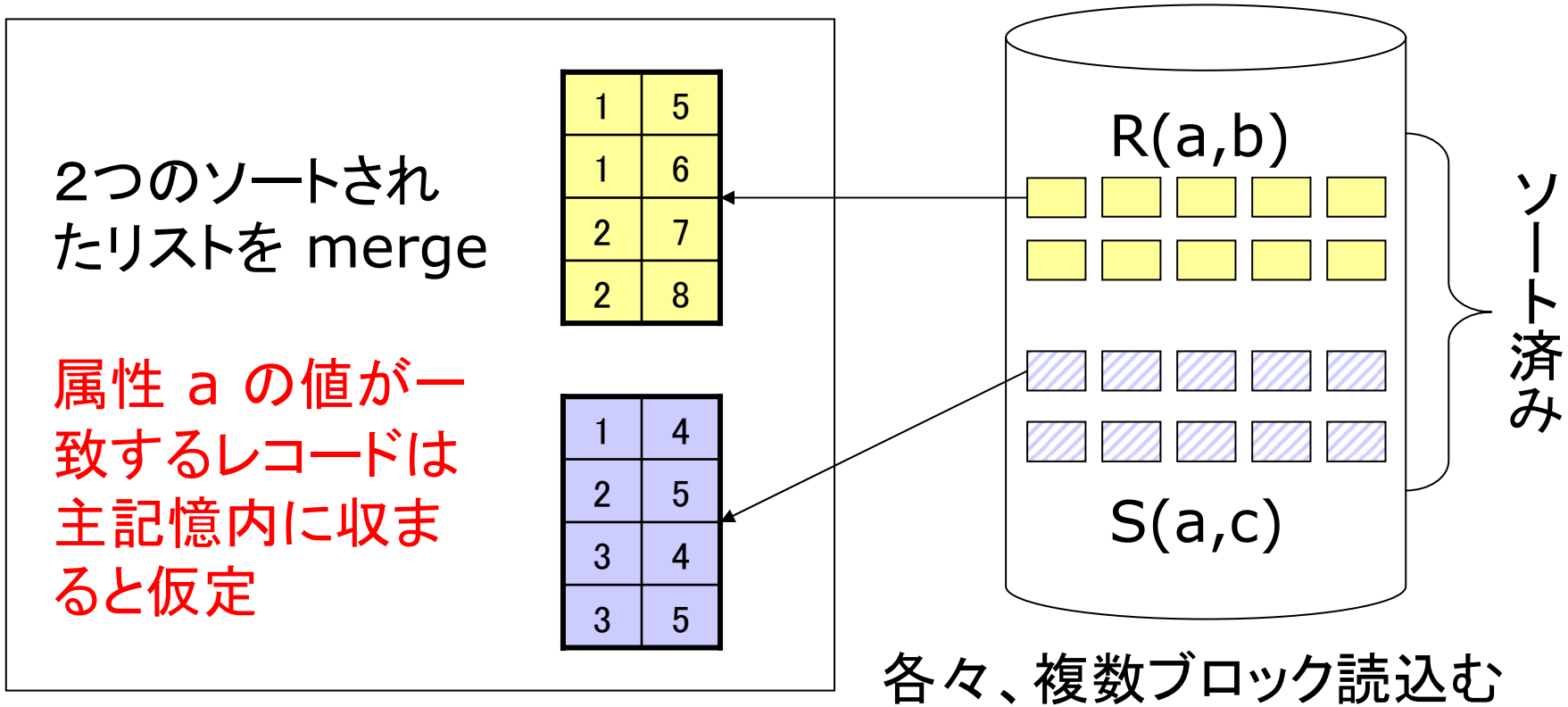
name	district	price
6	Yokohama	550
1	Tokyo	650
9	Chiba	450
3	Tokyo	750
5	Yokohama	850
4	Tokyo	850
7	Yokohama	600
8	Chiba	600
2	Tokyo	550

ソートを使ったグループ分けと集約



ソートを使ったジョインの実装

R(a,b) ▷◁ S(a,c)
まず R(a,b) および S(a,c) を属性 a で一気にソートしてしまう
4B(R)+4B(S) 個のブロック転送



各々、複数ブロック読込む
B(R)+B(S) 個ブロック転送

ソートを使ったジョインの実装

効き目がない最悪のデータ
属性 a の値がすべて同じ

Nested-Loop Join で対応

R(a,b)

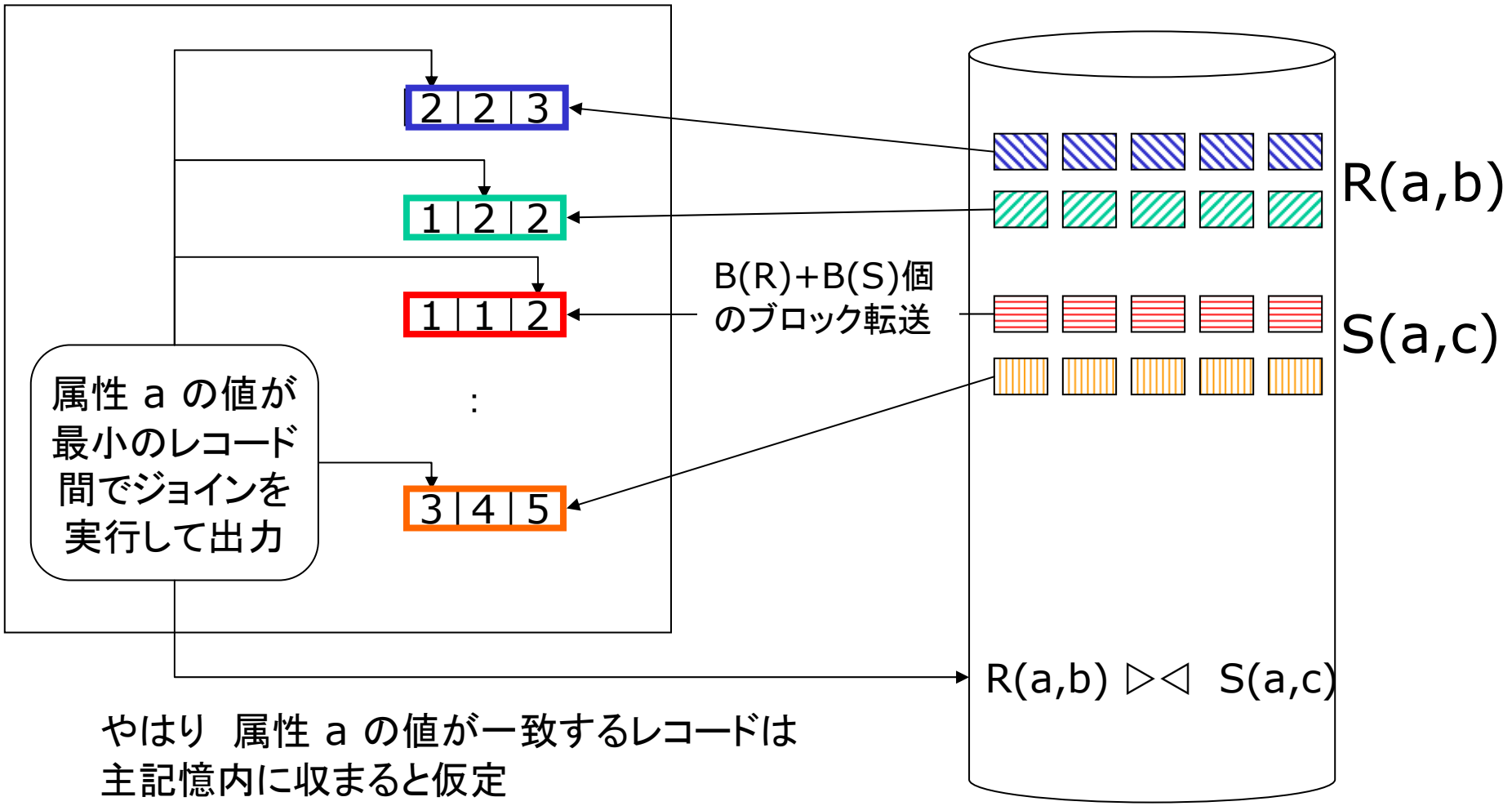
1	1
1	2
1	3
1	4
1	5
1	6

S(a,c)

1	1
1	2
1	3
1	4
1	5
1	6

ソートを使ったジョインの実装（改良版）

M 個のブロック毎に属性 a でソート (全体をソートしない)
2B(R)+2B(S) 個のブロック転送で済ましておく



やはり 属性 a の値が一致するレコードは
主記憶内に収まると仮定

問題

∪、∩、－　を実装する際に、各データの頻度表を主記憶内に格納できれば、二次記憶中のデータを1回だけ走査する方法は述べた。

頻度表が主記憶に入らないほど大きなデータを処理する場合に、ジョインを実装した方式のようにソートを使って ∪、∩、－　を実装する方式を述べよ。

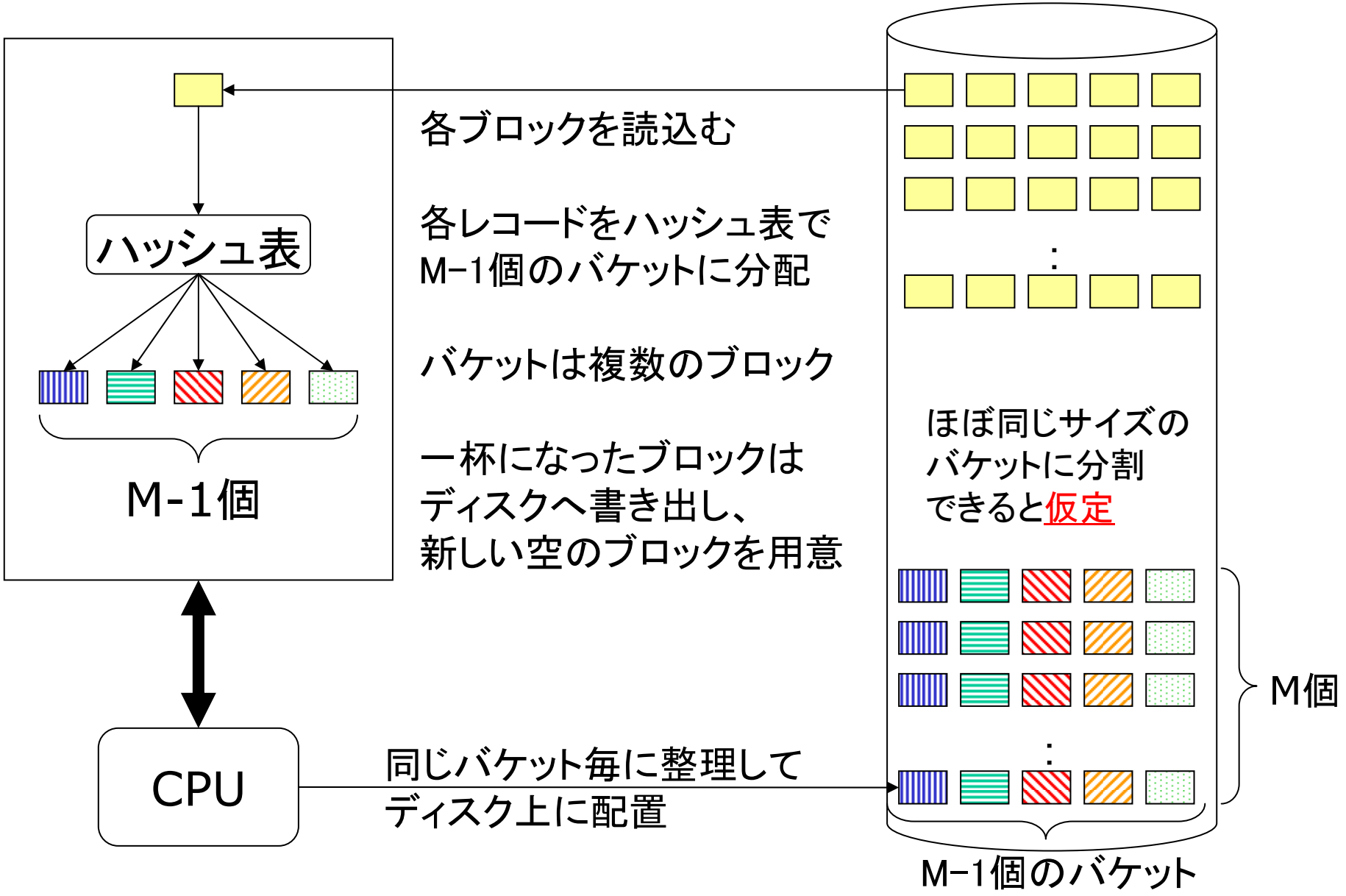
問合せの代数の実装

DBを2回走査することで
より大きな DB を処理できる方法

ハッシュ関数をつかう場合

問合せの並列化

ハッシュ関数をつかって関係を分割

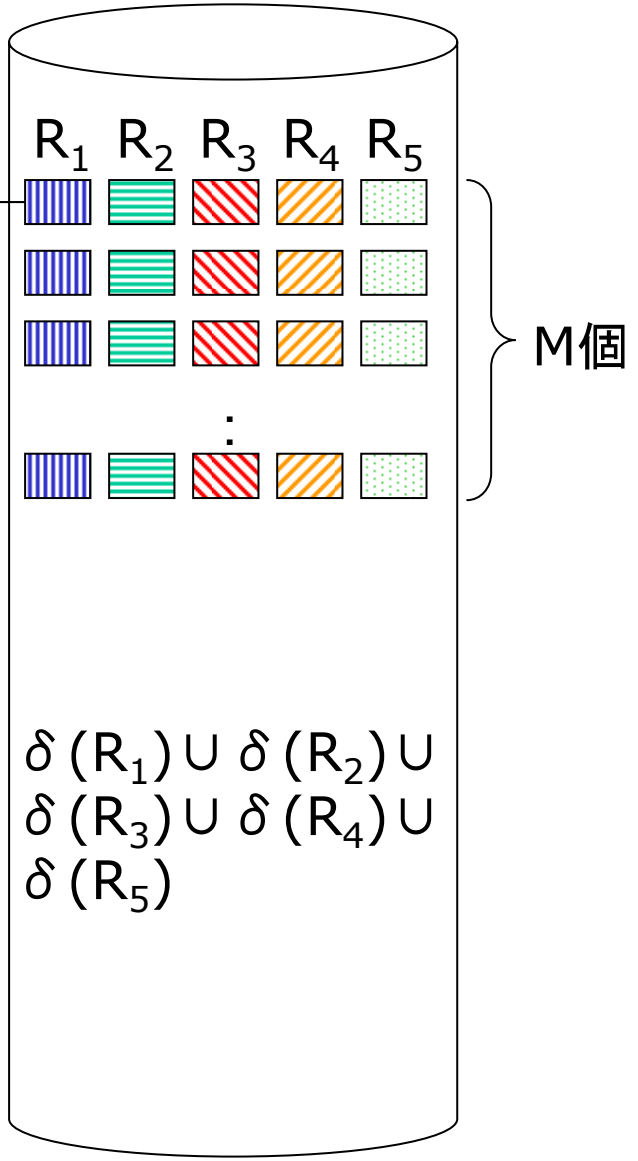


ハッシュをつかった重複除去

同じ値をもつレコードは必ず
同じバケットにあると仮定

（むろん同じ値をもつレコードがバケットに入らないほどあれば、この仮定は崩れる。ソートをつかった方法を利用すべき）

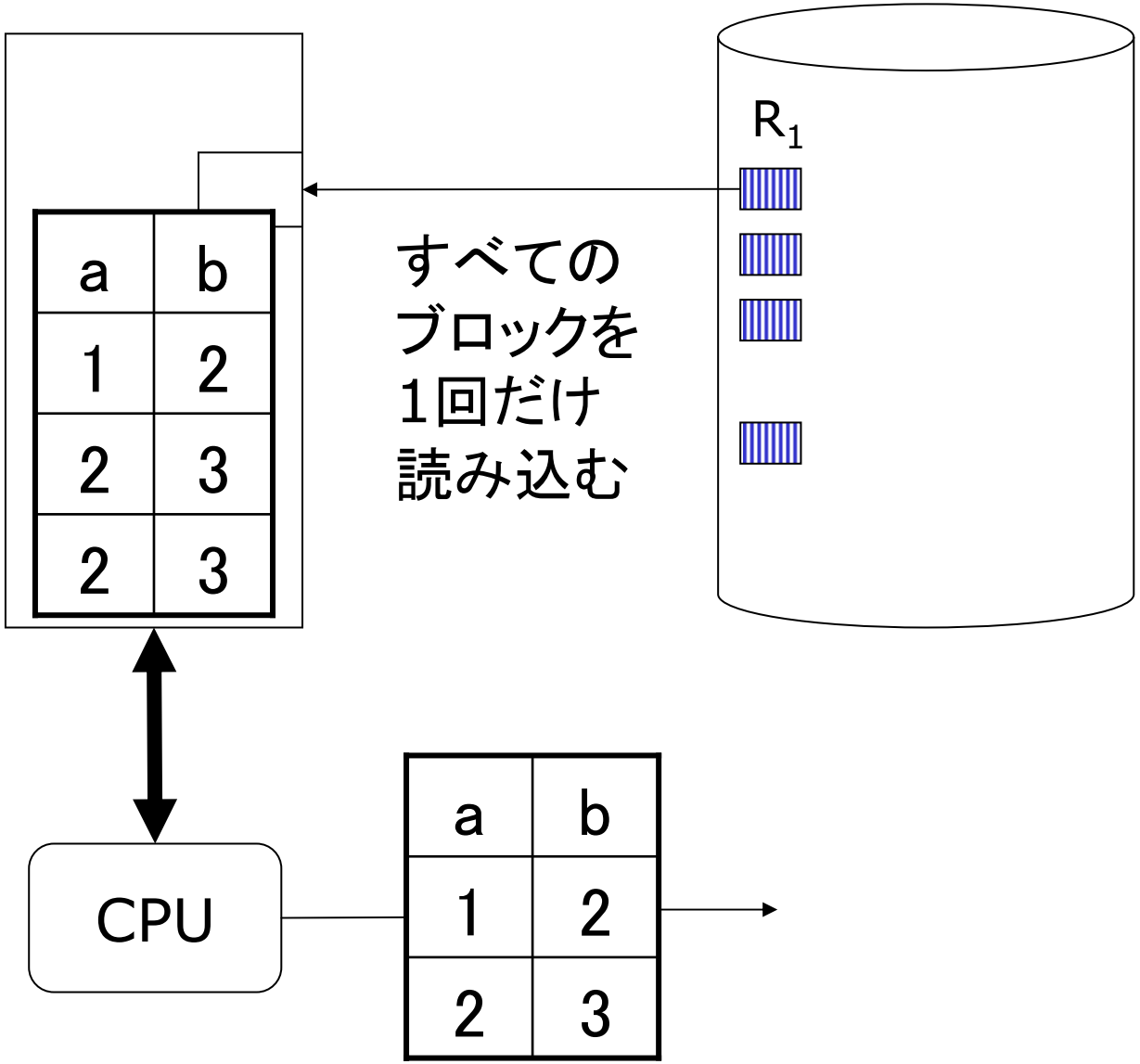
バケット毎に独立して
重複除去を実行できる
DBを一回だけ走査する
重複除去を各バケットに
対して実行



DBは1回だけ走査する重複削除を各バケットに適用

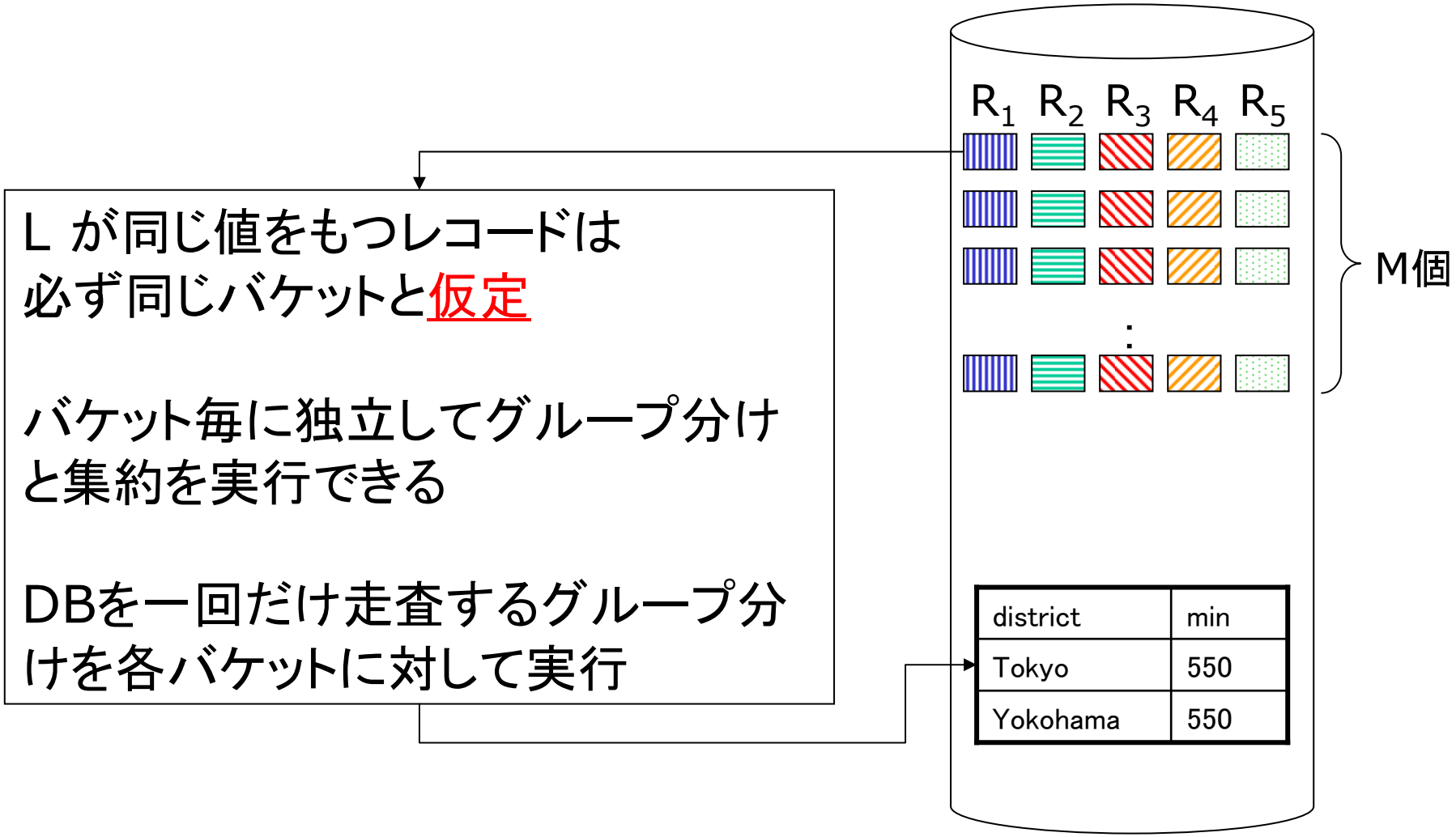
十分な主記憶があり
既に読んだレコードか
否かの判定を高速に
実行できる場合

ハッシュ表や
平衡2分木で
管理できる



ハッシュをつかったグループ分けと集約

$\gamma_L(R)$　　グループ分けの対象となる属性 L の値をつかいハッシュ関数で分割しておく



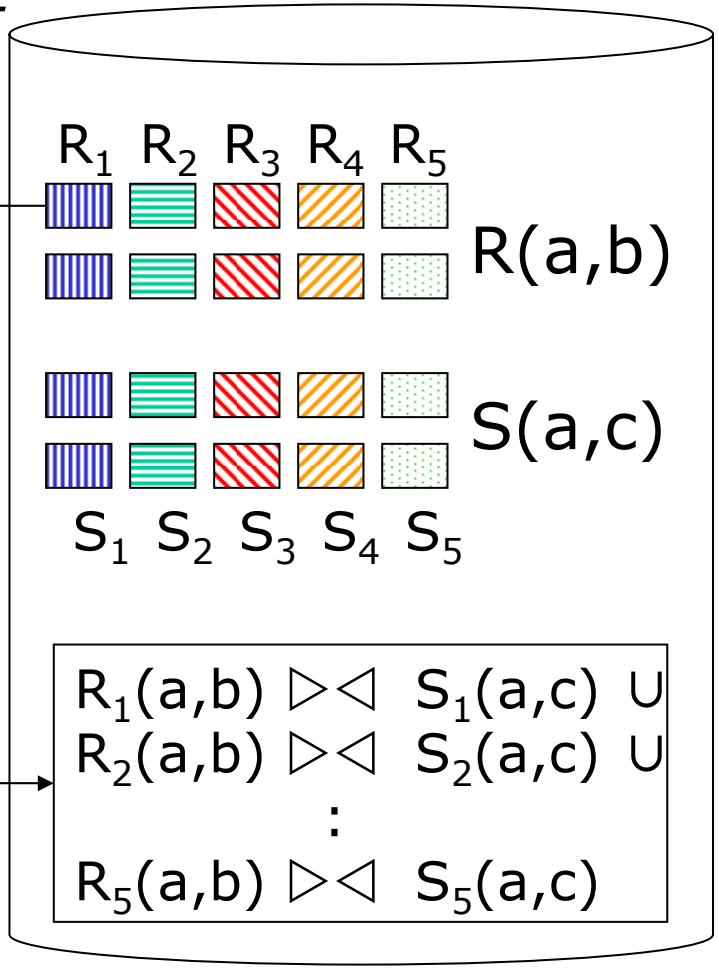
ハッシュ・ジョイン

R(a,b) ▷◁ S(a,c)
まず R(a,b) および S(a,c) を属性 a の値によりハッシュで分割
2B(R)+2B(S) 個のブロック転送

属性 a が同じ値をもつレコードは
必ず同じバケットと仮定

バケット毎に独立してジョインを実行

DBを一回だけ走査するジョインを
R_i と S_i の組に対して実行し和をとる

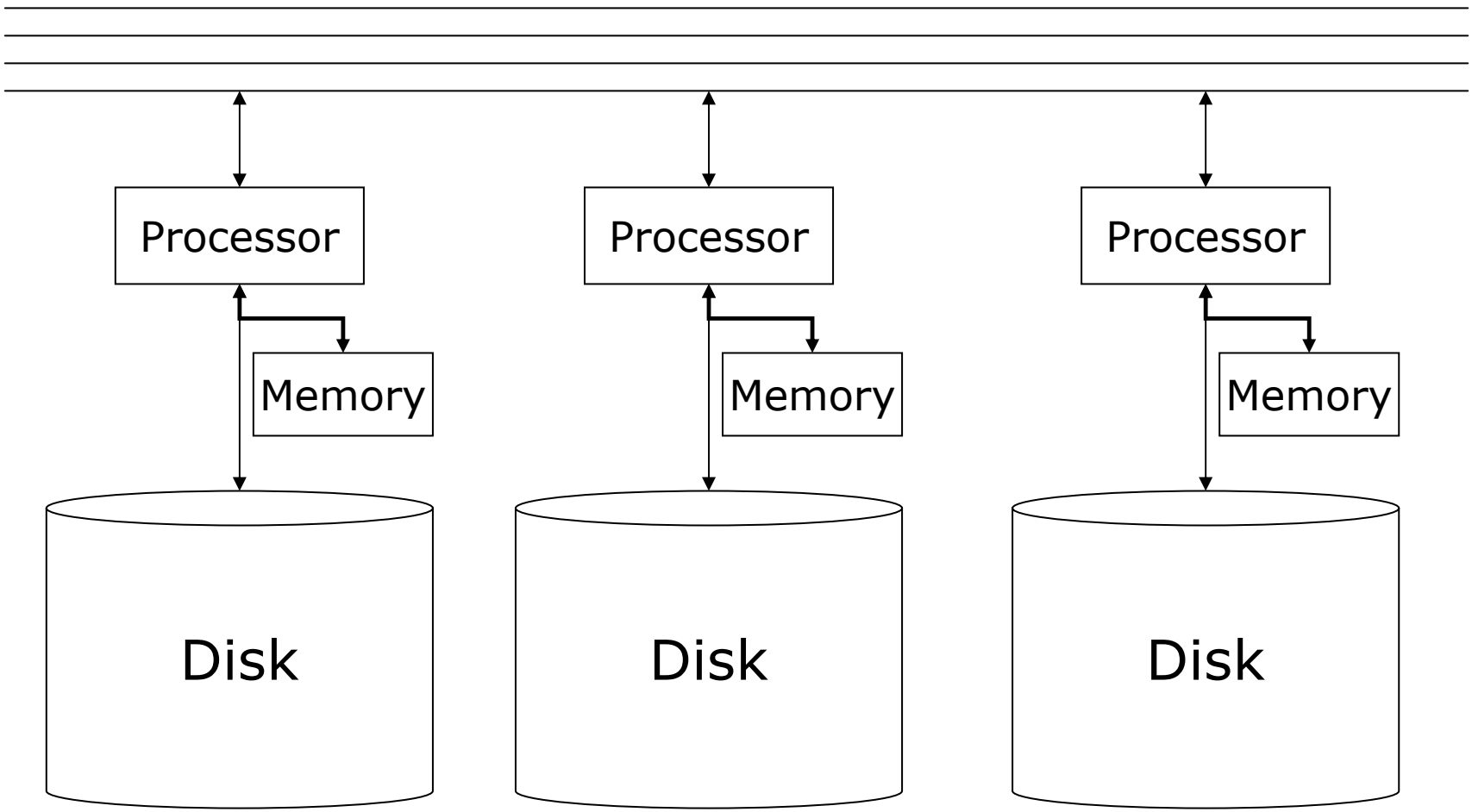


問題

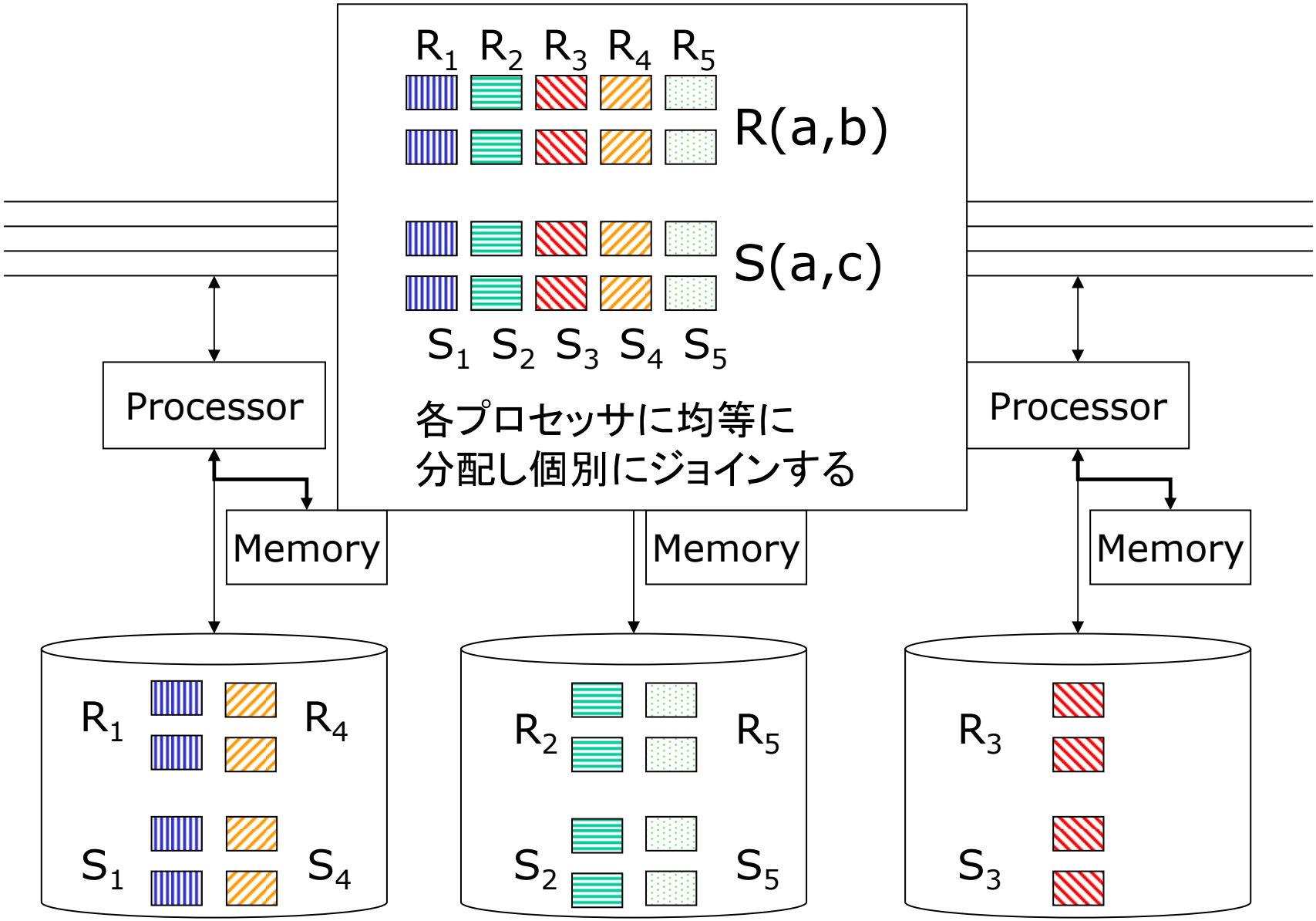
頻度表が主記憶に入らないほど大きなデータを処理する場合に、ハッシュ表を使って ∪、∩、－　を実装する方式を述べよ。

Shared-Nothing 型並列計算機でのハッシュ・ジョイン

DB 問合せは、Shared-Nothing 型並列計算機（たとえば Blade クラスター）
での実行になじむため Shared-Nothing 型はデータベース用計算機として標準的



Shared-Nothing 型並列計算機でのハッシュ・ジョイン

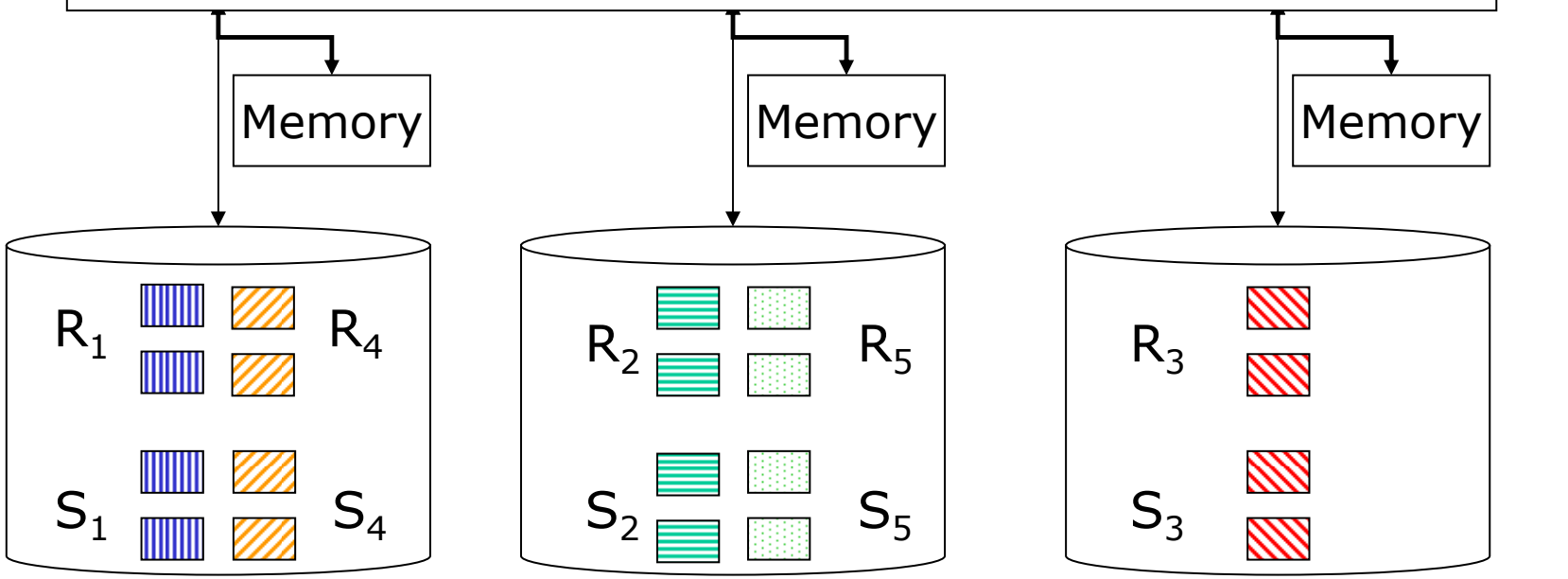


Shared-Nothing 型並列計算機でのハッシュ・ジョイン

p 個のディスクへ分配し、入出力を並列化することで性能を向上
均等に分配された後は、各プロセッサは1回の入力で
(B(R)+B(S))/p ブロックだけ転送すればよい

例

B(R)=1000, B(S)=500, プロセッサ数=10
各プロセッサに 150ブロック



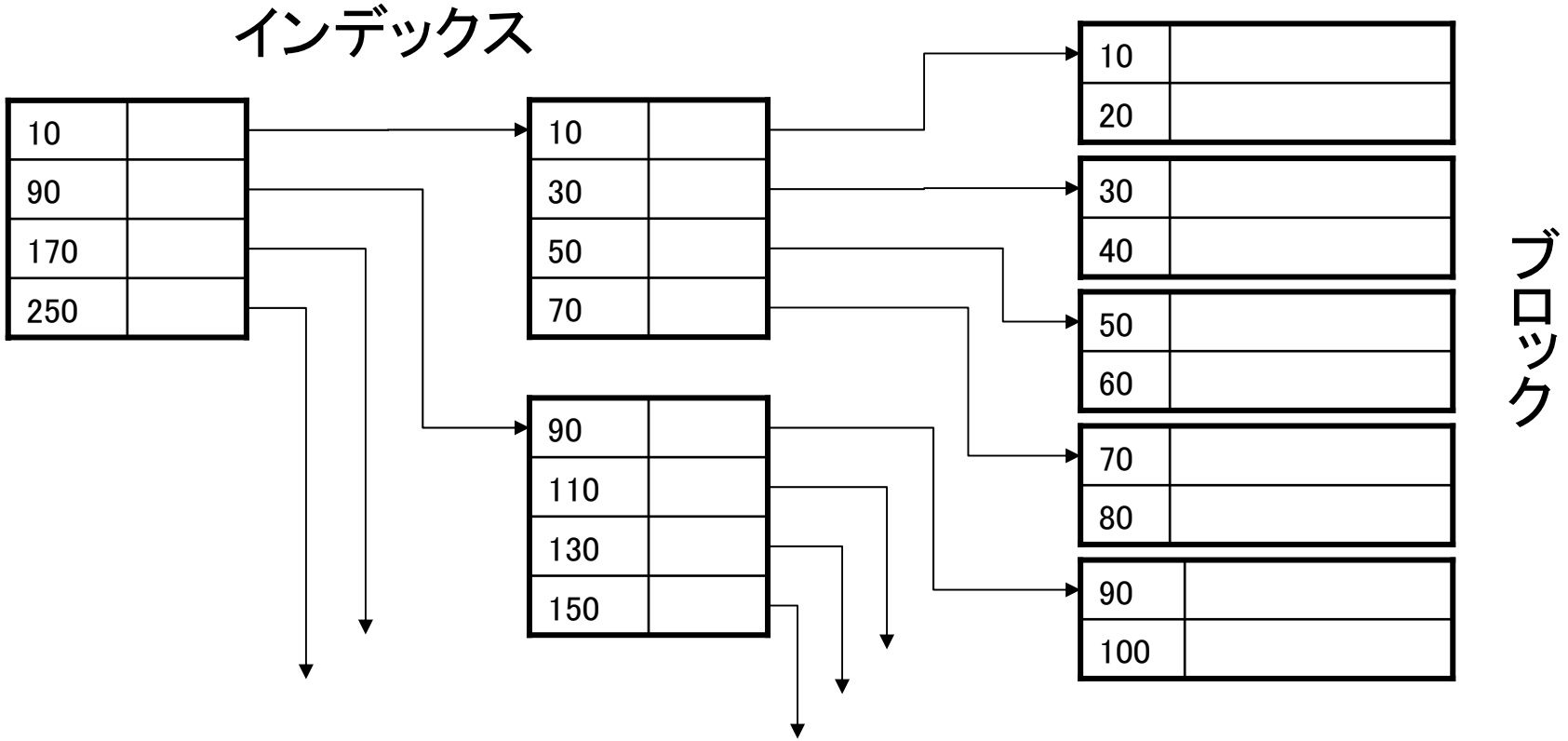
問合せの代数の実装

インデックスが利用できる場合

インデックス

インデックス: 属性 a の値が v となるレコードが
どのブロックにあるか、おおよその位置を保持した表

例えば、識別子(ID) の順番にレコードをディスク上に格納
インデックスは各 ID が格納されたブロックのおおよその位置を保持



その他のインデックス

- ２次的インデックス
  複数の属性にインデックス （例　名前と電話番号）
- B-木 (B-trees)
- ハッシュ表

技術的工夫

1. 更新への対応
2. 大量のデータにもアクセス速度を劣化させない

詳しくは　参考教科書の 4 章を参照

H. Garcia-Morina, J. D. Ullman, and J. Widom. Database System Implementation. Prentice Hall, 2000, ISBN 0-13-040264-8

インデックスを利用した問合せの高速化

$\sigma_{a=v}(R)$

属性 a にインデッスクがある場合
⇒ インデックスをつかった高速なアクセス

$R(a,b) \triangleright\triangleleft S(a,c)$

S(a,c) の属性 a にインデッスクがある場合
⇒ R(a,b) の各レコード毎に
インデックスをつかった問合せを実行

問合せ速度を改善するための代数的操作

交換律と結合律

$R \times S = S \times R$ $\quad\quad$ $(R \times S) \times T = R \times (S \times T)$

Natural Join

$R \bowtie S = S \bowtie R$ $\quad\quad$ $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$

$R \cup S = S \cup R$ $\quad\quad$ $(R \cup S) \cup T = R \cup (S \cup T)$

$R \cap S = S \cap R$ $\quad\quad$ $(R \cap S) \cap T = R \cap (S \cap T)$

$\cup$ と $\cap$ に関する交換律と結合律は、bags 意味論と集合意味論のどちらでも成立

代数的操作がうまくゆかない例

集合意味論では成り立つ分配律が bags 意味論だと不成立

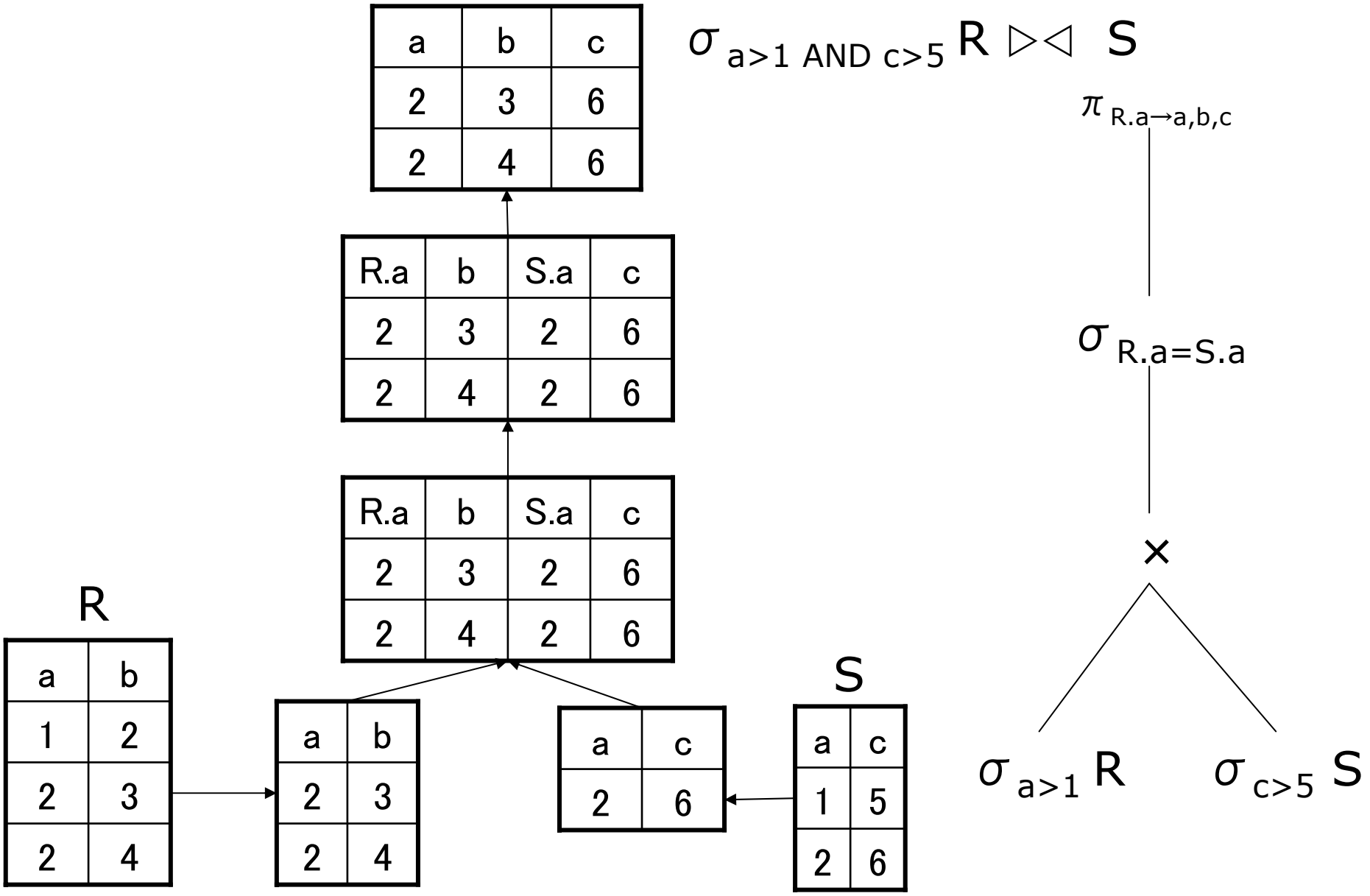
- R=S=T={a}
- R ∩ (S ∪ T) = {a} ∩ {a,a} = {a}
  (R ∩ S) ∪ (R ∩ T) = {a} ∪ {a} = {a,a}

Theta-Join では 結合律が意味をもたな場合がある

R(a,b), S(b,c), T(c,d)
$(R \triangleright\!\triangleleft_{R.b > S.b} S) \triangleright\!\triangleleft_{a < d} T$

$R \triangleright\!\triangleleft_{R.b > S.b} (S \triangleright\!\triangleleft_{a < d} T)$　　a は S と T の属性でない

選択を先行実行して効果がある例



選択を先行実行するための操作　AND, OR

$\sigma_{C1\ AND\ C2}(R) = \sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R))$

C1 と C2 どちらを先に実行してもよい

R に重複したレコードが存在しない場合

$\sigma_{C1\ OR\ C2}(R) = \sigma_{C1}(R) \cup_{set} \sigma_{C2}(R)$

ただし　$\cup_{set}$ は集合和

R が重複したレコードが存在する場合

$\sigma_{C1\ OR\ C2}(R) = \sigma_{C1}(R) \cup \sigma_{C2}(R)$

は必ずしも成り立たない

$\cup$ は bag 意味論であることに注意

問題

Bags 意味論において、以下の関係が成立しない反例と、成立する場合を示せ。

$$\sigma_{C1\ OR\ C2}(R) = (\sigma_{C1}(R)) \cup (\sigma_{C2}(R))$$

選択を先行実行するための操作　∪、－、×、▷◁、∩

$\sigma_C(R \cup S) = \sigma_C(R) \cup \sigma_C(S)$

$\sigma_C(R - S) = \sigma_C(R) - \sigma_C(S) = \sigma_C(R) - S$

以下、条件 C に現れる属性は、すべて R の属性と仮定

$\sigma_C(R \times S) = \sigma_C(R) \times S$

積 × の場合、R と S は共通の属性を含まない

$\sigma_C(R \bowtie S) = \sigma_C(R) \bowtie S$

C が R と S に共通する属性を含む場合でも成立

$\sigma_C(R \cap S) = \sigma_C(R) \cap S$

選択を先行実行するための操作　∪、－、×、▷◁、∩

例　　選択の先行実行

$\sigma_{(a=1 \text{ OR } a=3) \text{ AND } b<c}(R(a,b) \triangleright\triangleleft S(b,c))$

$\sigma_C(R \triangleright\triangleleft S) = \sigma_C(R) \triangleright\triangleleft S$　は使えない
条件が R(a,b) の属性以外に属性 c にも作用するから

$= \sigma_{a=1 \text{ OR } a=3}(\sigma_{b<c}(R(a,b) \triangleright\triangleleft S(b,c)))$
b<c を push

$= \sigma_{a=1 \text{ OR } a=3}(R(a,b) \triangleright\triangleleft \sigma_{b<c} S(b,c))$
b<c を さらに push

$= (\sigma_{a=1 \text{ OR } a=3} R(a,b)) \triangleright\triangleleft (\sigma_{b<c} S(b,c))$
a=1 OR a=3 は R(a,b) のみに作用

問題

以下の2つの性質が成立することを説明せよ（できれば証明せよ）

$\sigma_C(R - S) = \sigma_C(R) - \sigma_C(S) = \sigma_C(R) - S$

$\sigma_C(R \triangleright\triangleleft S) = \sigma_C(R) \triangleright\triangleleft S$

条件 C に現れる属性は、すべて R の属性と仮定

射影の先行実行

- 選択の先行実行
  計算途中で生成されるレコードの数を減らす効果

- 射影の先行実行は、効果的か？
  表の幅を狭くするが、レコードの数を減らさない
  （bags 意味論の場合）
  選択の先行実行ほどの効果はない

- 集合意味論の場合
  射影すると重複レコードを除ける場合がある

射影を先行実行するための操作

ジョイン等を実行するのに必要な属性と被射影属性を残す

- $\pi_L(R \triangleright\triangleleft S) = \pi_L(\pi_M R \triangleright\triangleleft \pi_N S)$
  M: S もしくは L に現れる R の属性全体
  N: R もしくは L に現れる S の属性全体

$\pi_{a+e\rightarrow x,b\rightarrow y}(R(a,b,c,d) \triangleright\triangleleft S(c,e,f))$
$= \pi_{a+e\rightarrow x,b\rightarrow y}(\pi_{a,b,c}R(a,b,c,d) \triangleright\triangleleft \pi_{c,e}S(c,e,f))$

- $\pi_L(R \times S) = \pi_L(\pi_M R \times \pi_N S)$
  M = L に現れる R の属性全体
  N = L に現れる S の属性全体

射影を先行実行するための操作

- $\pi_L(R \cup S) = \pi_L(R) \cup \pi_L(S)$
- $\pi_L(R \cap S) = \pi_L(R) \cap \pi_L(S)$ は必ずしも成り立たない
  （$\cap$ は bags 意味論）
- $\pi_L(\sigma_C(R)) = \pi_L(\sigma_C(\pi_M(R)))$

  M は L もしくは C に現れる R の属性

問題

- $\pi_{a,f}(R(a,b,c,d) \triangleright\triangleleft S(c,d,e,f))$ において
射影を先行実行する式を示せ

- $\pi_L(R \cap S) = \pi_L(R) \cap \pi_L(S)$ が成立しない例を示せ
（$\cap$ は bags 意味論）

重複除去を先行実行するための操作

重複除去によりレコード数が減ることを期待して

重複除去を先行して実行する

$$\delta(R \times S) = \delta(R) \times \delta(S)$$

$$\delta(R \triangleright\triangleleft S) = \delta(R) \triangleright\triangleleft \delta(S)$$

$$\delta(R \triangleright\triangleleft_C S) = \delta(R) \triangleright\triangleleft_C \delta(S)$$

$$\delta(\sigma_C(R)) = \sigma_C(\delta(R))$$

$$\delta(R \cap S) = \delta(R) \cap \delta(S)$$

問題

成立しない性質　反例を示せ

$\delta(R \cup S) = \delta(R) \cup \delta(S)$

$\delta(R - S) = \delta(R) - \delta(S)$

$\delta(\pi_a(R(a,b))) = \pi_a(\delta(R(a,b)))$

尚、集合意味論ではこれらの性質は成立

なぜならレコードの重複を許さない意味論だから