DEIM Forum 2021 B11-3

世代管理されたマスタ表と トランザクション表との結合処理に関する考察

高田　実佳† 　合田　和生† 　喜連川　優†

† 東京大学生産技術研究所　〒 153–8503 東京都目黒区駒場 4-6-1
E-mail: †{mtakata,kgoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　様々なドメインでビッグデータを蓄積・分析への期待が高まっている．そうした分析対象データには刻一刻と増加するトランザクション表と，複数のマスタ表が含まれ，分析者はトランザクション表のデータを正しく理解する為に，適切なマスタ表を参照する必要がある．その為，分析者は適切なマスタ表とトランザクション表を結合処理した結果を必要となるが，データの運用期間が長くなるにつれ，マスタ表が更新を繰り返すと，トランザクション表との結合処理が複雑になりその処理時間が増大する．本論文ではそうした世代管理されたマスタ表とトランザクション表の結合処理方式を検討し初期評価について述べる．

キーワード　データベース，マスタ表，結合処理

1 はじめに

社会問題が複雑化する中，実世界とサイバー空間を融合させ，実世界の様々な情報を IoT によってサイバー空間に収集・蓄積し，AI などを用いて分析・判断することで実世界にフィードバックさせる，超スマート社会「Society 5.0」への期待が高まっている [1]．デバイスの発展に伴い，様々な分野で日々データの収集は進み，分析に活用できるデータ量は年々増加している [2]．特に，日本の医療分野では，超高齢化社会の到来により増え続ける医療費の軽減を目指して，医療費適正化計画の作成・実施・評価のための分析に使うため，ナショナルデータベース (NDB) を構築・運用しており，平成 30 年までにレセプトデータは約 148 億件以上，特定健診データは約 3 億件以上蓄積されている [5]．現在 NDB はほぼ全国民のデータを有しており，国民の医療・歯科診療・検診に関するデータを用いた悉皆性分析に有効であるいわれている [6]．こうした日々医療行為が行われる毎に増加するレセプトデータは，トランザクションデータと呼ばれる．一方で，マスタデータと呼ばれるトランザクションデータを構成する情報の基礎情報があり，両者を照らし合わせることでデータの中身を正しく理解することができる．トランザクションデータが増え続けるのに対し，マスタデータは頻繁には増加・修正は起こらないが，数ヶ月に 1 回程度〜数年に 1 回といった頻度で更新することが知られており，マスタデータの更新は既存値の変更，既存分類の変更，項目の追加，などドメインやタイミングによって様々である．この様に，マスタデータは世代によって更新し続けるので，トランザクションデータを正しく理解する為にはトランザクションデータの記録された時点の適切なバージョンのマスタデータを参照する必要がある．そのためマスタ表とトランザクション表との結合処理が必要となるが，マスタ表の更新内容や更新時期を考慮したトランザクション表と結合は複雑になり，マスタ表の世代が多くなくなるにつれその結合処理の性能が課題となる．本論文では医療分野に注目し，第 2 章で社会保険診療報酬支払基金のデータと電子カルテデータにおけるマスタデータの更新，およびそれに伴うトランザクションデータを関係データベースを用いて管理・検索するユースケースを整理し，課題を述べる．その課題に対し，著者らが提案する，世代管理マスタアウェア結合方式し，その効果を検証・考察し，将来に向けた課題を纏める．

本論文の構成は，以下の通りである．第 2 章では，社会保険診療報酬支払い基金の医療レセプトデータにおけるケーススタディを説明し，課題について言及する．第 3 章では，提案する複数世代マスタアウェア結合方式について説明する．第 4 章では，複数世代マスタアウェア結合方式の実験における評価を示す．第 5 章では，関連文献について言及し，第 6 では，今後に向けた課題と将来展望について纏める．

2 課　　題

2.1 医療データにおけるケーススタディ

社会保険診療報酬支払基金を通して蓄積されたレセプトデータ，特定健診データの基本となるマスタデータには，医科診療行為マスタ，歯科診療行為マスタ，調剤行為マスタ，医薬品マスタ，特定器材マスタ，コメントマスタ，傷病名マスタ，修飾語マスタ，歯式マスタ，の 9 種類がある [8]．それぞれ改訂回数も時期も様々であり各マスタの改訂数は表 1 の通りである [4]．

例えば傷病名マスタは過去 10 年の間約 40 回の更新が記録されており，その改訂の一部を表 2.1，2.1 に示す．

表 2.1 には 2020 年 6 月以前，表 2.1 には 2020 年 6 月以後の傷病名マスタの改訂の様子を示している．2020 年以前も以後も表スキーマは変更なく．各カラムの意味は次の通りである．変更区分はマスタ内容の異動状況を表し，9 は廃止を意味する [8]．マスタ種別は B で傷病マスタであることを表す．傷病名コードはレセプト電産処理システムの診療報酬請求用に使用するコー

表 1　レセプトマスタの改訂

マスタ種類	改訂回数	改訂時期
医科診療行為マスタ	31	2020/3/5-2020/12/18
歯科診療行為マスタ	18	2020/3/5-2020/12/16
調剤行為マスタ	5	2020/3/5-2020/12/16
医薬品マスタ	13	2020/3/5-2020/12/10
特定器材マスタ	12	2020/3/5-2020/12/17
コメントマスタ	7	2020/3/27-9/1
傷病名マスタ	40	2010/10/1-2020/6/1
修飾語マスタ	25	2010/3/1-2020/6/1
歯式マスタ	1	2011/10/3

ドであり，移行先コードは傷病名の異動先の傷病名コードを表す．これは，変更区分が 9 であり廃止が決定している傷病名コードに対して新しい傷病名コードがあれば移行先コードが付与されていることを示す．傷病名基本名称は傷病名の表記標準化の基本名称である．例えば，2020 年 6 月以前の傷病名マスタでは，旧傷病名コード 19002 の仮性小児コレラはその傷病コードが廃止となり，次に 88005 の傷病コードになることが決まっている事を表している．そして，2020 年 6 月以降の傷病名マスタでは，傷病名コード 88005 にはロタウイルス性感染炎，という傷病名基本名称となっている．同様に，細菌性赤痢やぶどう球菌性食中毒，についても傷病名コードが変更になり，食あたりや胃カタルなどは 2020 年 6 月で旧傷病名コードは廃止となっていることが分かる．ぶどう球菌性食中毒，についても傷病名コードが変更になり，食あたりや胃カタルなどは 2020 年 6 月で旧傷病名コードは廃止となっていることが分かる．

一方，表 4 にトランザクションデータである医療レセプトレコード例を示す [7]．各レコード ID に対して，傷病名コードは傷病名マスタに記載されている傷病名コード，医療機関で受診した診療回数と診療開始日，請求年月がある場合，患者数や訪問医療機関，年月が経つにつれ本医療レコード表は増加する．

医療レセプトには傷病コードがあるがそのコードが意味する傷病名データないので，傷病名を把握する為には医療レセプトとマスタ表を結合する必要がある．下記に示す検索クエリ例を用いて，結合処理を行った場合の出力結果を表 5 を示す．本結合では，複数ある傷病マスタ表と医療レセプトレコードを傷病名コードをキーに結合する．

```
(SLECT␣ID. 傷病名コード, 傷病名基本名称, 診療回数
FROM␣傷病名マスタ 1␣R1, 医療レセプト␣S
WHERE␣R1. 傷病名コード=S. 傷病名コード
)UNION(
SLECT␣ID. 傷病名コード, 傷病名基本名称, 診療回数
FROM␣傷病名マスタ 2␣R2, 医療レセプト␣S
WHERE␣R2. 傷病名コード=S. 傷病名コード
)UNION(
...
)UNION(
SLECT␣ID. 傷病名コード, 傷病名基本名称, 診療回数
FROM␣傷病名マスタ m␣Rm, 医療レセプト␣S
WHERE␣Rm. 傷病名コード=S. 傷病名コード
);
```

このようなクエリの場合，各マスタ表とトランザクション表をそれぞれ結合し，最後に併合しており，トランザクション表の選択条件がなく，選択率が高く全データを読み込む場合が生じている．これを従来方式 I とすると，この場合，マスタ表が m 個存在する時には m 個のマスタ表とトランザクション表とのハッシュ結合を実行する．その後，選択条件があれば重複を排除する．この方式では本来有効ではないマスタ表とトランザクション表との結合が生じ，不必要なハッシュ結合を実行する為，m が多くなるにつれてその結合にかかる処理時間が大きくなる．

また，別の結合方式 (従来方式 II) には，複数のマスタ表を最初に共通項目を抽出し併合したマスタ表を生成した後にトランザクション表と結合と行う処理方式も実行可能であるが，ユーザが多くの世代のマスタ表の中から適切なマスタ表を選択し複雑なクエリを指定する必要に加え，複数のマスタ間に差分がある程不必要な結合が生じることになる．

3　複数世代マスタアウェア結合方式

本論文では，複数世代のマスタ表が存在する時にナイーブなハッシュ結合に要するコストを削減する複数世代マスタウェア結合方式を提案する．これは，マスタ表の登録情報をメタデータとして管理しておくことにより，本来結合する必要のないマスタ表とトランザクション表の結合が不必要に実行することを避け，トランザクション表に対して適切な範囲だけを読み込みをすることを特徴とする結合方式である．

Algorithm 1 Multi-version-aware Join

Require: $R_x(x = 1, 2, .., m)$ and S

1: Prepare $f \leftarrow \{R_1, R_2, \cdots, R_m\}$
2: Build $\{\hat{R_1}, \hat{R_2}, \cdots, \hat{R_m}\} \leftarrow \{R_1, R_2, \cdots, R_m\}$
3: **for** $s \in S$ **do**
4: 　　$f(s) \rightarrow i$
5: 　　Probe $s \rightarrow \hat{R_i}$
6: **end for**

その概略を図 1 に示す．ここで，m 個のバージョンが存在するマスタ表 $R_x(x = 1, 2, 3, .., m)$ とし，トランザクション表を S 表，とした時の本提案方式をアルゴリズム 1 に示す．

まず初めに，m 個のマスタ表群 R_x が，$x = 1, 2, 3, .., m$ からハッシュテーブルを生成しておく．そしてトランザクション表 S の各レコードに対して，有効なマスタ表 R_i を選択し，選択したマスタ表とのみプローブする．これを S の全レコードに対して実行する．これより，本提案方式は従来方式 I，従来方式 II と比べて不必要な結合を要しないことで結合処理の回数を低減することができる．トランザクション表行数が各マスタ表の行数に比べて圧倒的に大きいと想定すると，本提案方式は最大約 $1/m$ の結合処理回数を削減することができると推定できる．

本提案方式を第 2.1 章で示した傷病名マスタ表とレセプトのトランザクション表との結合に適応した場合の結合例を図 2 に

表 2　傷病名マスタ:2020-06 以前

変更区分	マスタ種別	傷病名コード	移行先コード	傷病名基本名称
9	B	19002	88005	仮性小児コレラ
9	B	49003	49005	細菌性赤痢
9	B	50001	50002	ぶどう球菌性食中毒
9	B	59008	59003	食物中毒
9	B	85001	85004	急性細菌性腸炎
9	B	90008	90003	腸管感染症
9	B	90010	90001	急性感染症胃腸炎
9	B	90012	90003	腸内感染
9	B	90013	90003	腸管感染
9	B	90014	90003	急性感染性腸炎
9	B	91001		うっ血性胃腸炎

表 3　傷病名マスタ:2020-06 以後

変更区分	マスタ種別	傷病名コード	移行先コード	傷病名基本名称
0	B	49005	49005	赤痢
0	B	50002	50002	ぶどう球菌食中毒
0	B	52004	52004	ウェルシュ菌食中毒
0	B	59003	59003	食中毒
0	B	62001	62001	アメーバ性非赤痢性大腸炎
0	B	85003	85003	細菌性大腸炎
0	B	85004	85004	細菌性腸炎
0	B	85005	85005	細菌性胃腸炎
0	B	88002	88002	ウイルス性胃腸炎
0	B	88005	88005	ロタウイルス性腸炎
0	B	90001	90001	感染性胃腸炎
0	B	90002	90002	感染性大腸炎
0	B	90003	90003	感染性腸炎

表 4　医療レセプトレコード例

ID	傷病名コード	医療機関コード	診療回数	診療開始日	請求年月
1	4905	101011	1	2020-04-02	2020-07
2	50002	102930	2	2020-04-02	2020-07
3	88004	20203	1	2020-04-02	2020-07
4	90001	202	3	2020-04-02	2020-07
5	90006	304043	2	2020-04-02	2020-07
6	91023	20291	2	2020-04-02	2020-07
…	…	…	…	…	…

表 5　複数マスタ表とトランザクション表の結合出力例

ID	傷病名コード	傷病名基本名称	診療回数
1	4905	赤痢	1
2	50002	ぶどう球菌食中毒	2
3	88004	ロタウイルス性腸炎	1
4	90001	感染症胃腸炎	3
5	90006	感冒性大腸炎	2
6	91023	腸炎	2
…	…	…	…

示す．レコード毎に適切なマスタ表との結合のみが選択され実行されることで結合処理時間の削減，および，ユーザのクエリ指定の複雑さを軽減すると予想でできる．

4　評　　価

4.1　実　　験

本実験では複数バージョンのマスタ表とトランザクション表の結合において，第 1 章で提案した複数世代マスタアウェア結合方式の優位性を従来方式 I と従来方式 II と比較することで示すため，TPC-H [25] のデータを用いた．本実験では dbgen を用

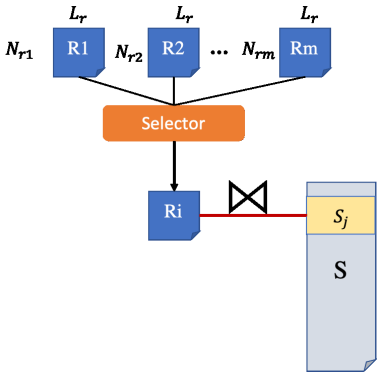

図 1　複数世代マスタアウェア結合方式

いてスケールファクタ 10 の生成し，生成した PART 表をベースとして 5% レコードが異なる複数の PART 表を生成した．各 PART 表は，そのカラムに持つ Retailprice の値が $(-1\%, 1\%)$ の範囲でランダムウォークで推移することを想定し生成し，生成した複数の PART 表と，LINEITEM 表の PARTKEY をキーとした結合について評価した．複数の PART 表毎の有効期間をカタログ表で管理し，LINEITEM 表の COMMITDATE に対して有効なマスタ表を選択するとした．実験機器には 56CPU, 96GB メモリ，31TB HDD を用いた．

4.2　結　　果

異なる結合対象となるマスタ表の数における従来方式 I，従来方式 II と提案方式におけるハッシュ結合の場合の応答時間を比較した．結果を図 3 に示す．これより，提案方式は結合対象マスタ表数が増加するにつれて結合性能が向上しており，結合対象マスタ表数が 12 個の場合，提案方式は従来方式 I より約 73%，従来方式 II より 66% 応答時間が向上してことが確認できる．

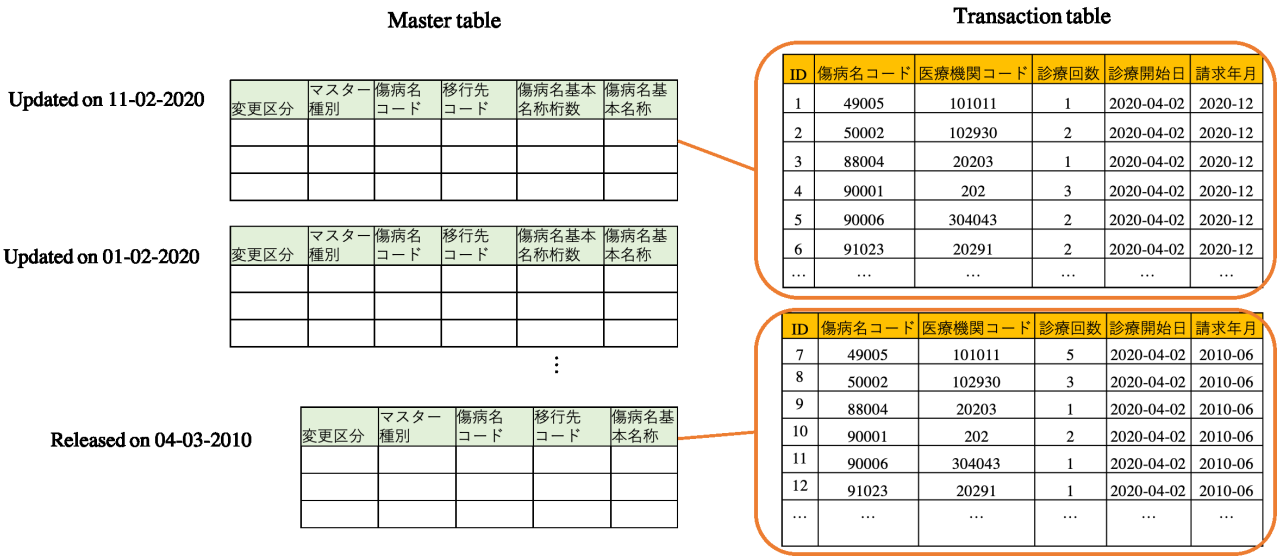

図 2　複数マスタ表とトランザクション表の結合イメージ

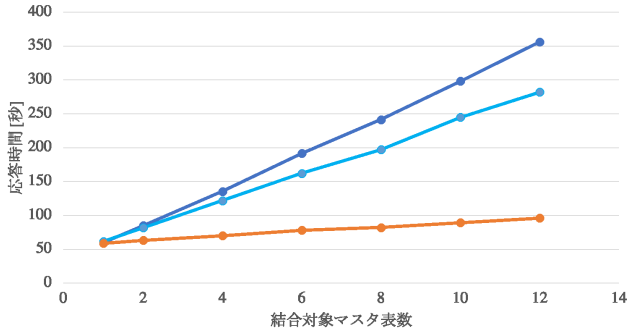

図 3　ハッシュ結合の応答時間

図 4 に各方式のハッシュ結合におけるビルド，カタログ読込，プローブの各時間を示しており，従来方式は結合対象マスタ表数が増加するにつれてプローブ時間が増加しているのに対し，提案方式は結合対象マスタ表数が増加してもプローブにかかる時間が一定であることから，全体の応答性能が向上していることを確認できる．

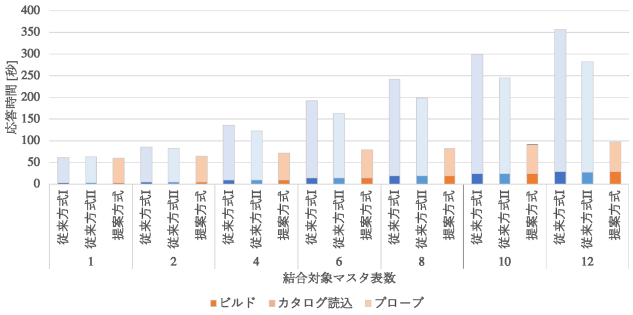

図 4　ハッシュ結合の応答時間詳細

従来方式 I，従来方式 II，提案方式の CPU 利用率を図 5-7，各 IO スループットを図 8-10 に示す．これより，本結合は CPU バウンドな処理であるが，従来方式 I，従来方式 I に比べ提案方式が CPU 利用率が低く，読込み量を向上できている事が分かる．

5　関 連 文 献

本研究は，分析等で頻繁に利用する処理をデータベース内にオフロードする事例はストアドプロシージャやオペレータなど長く研究されている [11] [12] [13]．統計や集計が分析で頻繁に行われることから使われるカラムストア [14] [15] [16] や，医療ユースケースで行われるコホート分析をデータベースにオペレーターとして処理をもたせることで，迅速にコホート分析を実現する研究 [24] がある．複数のデータを掛け合わせた分析において結合処理は必要不可欠なデータ処理であり，アプリケーションで実行するには負荷が高く，データベースエンジン内で処理を行うにしても，読み込み量や IO 数が増えるにつれ結合処理の負荷は大きくなる [17] [18]．こうした問題に対し，マスタ表とトランザクション表のハッシュ結合の代表的な手法にブルームフィルタがある [22]．これは，ハッシュ関数による変換値が集合に含まれるかどうかを判定することによって探索空間を限定することで，読み込みコストを抑制することができ，研究が行われてきている [19] [20] [21]．しかし，従来単表のマスタ表とトランザクション表との結合における高速化に適用されてきた．分散するデータとの多重結合におけるブルームフィルターが研究されているが，著者らが知る限り，第 2 章で述べたケースのように複数の世代のマスタ表との結合方式とは異なり，利用しないマスタ表がある場合を考慮していない [23]．

6　おわりに

本論文では，複数世代にまたがるマスタ表とトランザクションとの効果的な結合処理を提案し，その効果をハッシュ結合の場合において評価した．TPC-H のデータセットをベース生成した複数の PART 表と LINEITEM 表との結合において評価したところ，従来の結合方式に比べ最大 73% の応答時間の削減を確認できた．今後，索引結合における提案手法の優位性，

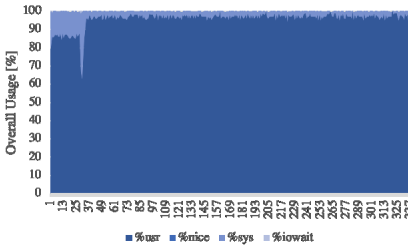

図 5　従来方式 I の CPU 利用率

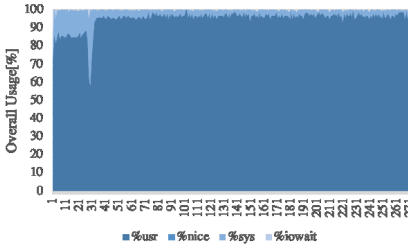

図 6　従来方式 II の CPU 利用率

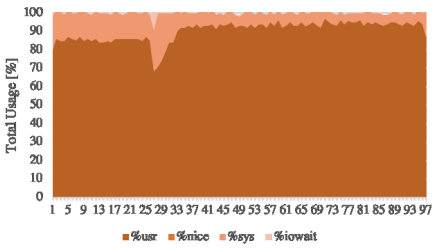

図 7　提案方式の CPU 利用率

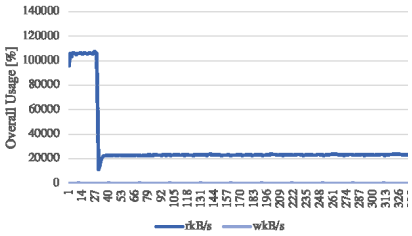

図 8　従来方式 I の IO

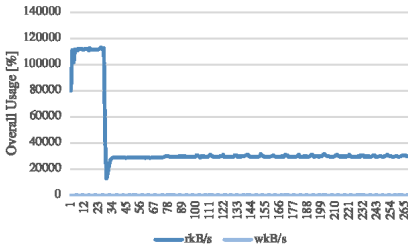

図 9　従来方式 II の IO

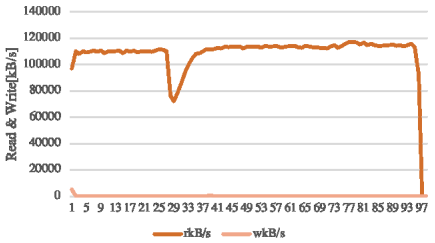

図 10　提案方式の IO

および第 2.1 章で解説した医療分野をはじめとしたリアルデータを用いた場合の有効性を示していきたい.

文　　献

[1] 内閣府. "Society5.0." https://www8.cao.go.jp/cstp/society5$_0$/ (参照 2020 − 12 − 21).

[2] Kambatla, Karthik, et al. "Trends in big data analytics." Journal of parallel and distributed computing 74.7 (2014): 2561-2573.

[3] Tsai, Chun-Wei, et al. "Big data analytics: a survey." Journal of Big data 2.1 (2015): 1-32.

[4] 基本マスター. 社会保険診療支払基金. 2020-12-03 更新. https://www.ssk.or.jp/smph/seikyushiharai/tensuhyo/kihonmasta/index.html, (参照 2020-12-21).

[5] 関総研グループ. "医療需要の実態把握に活用　NDB オープンデータの概要." 医業経営情報 REPORT (2018). http://www.sekisoken.co.jp/wp-content/uploads/2018/10/Report201810$_0$2.*pdf*, (参照 2020 − 12 − 20).

[6] 藤森研司. "レセプトデータベース (NDB) の現状とその活用に対する課題." 医療と社会 26.1 (2016): 15-24.

[7] 福田治久, 他. "NDB 解析用データセットテーブルの開発." 保健医療科学 68.2 (2019): 158-167.

[8] 社会保険診療報酬支払基金 編集. "レセプト電産処理システム マスターファイル仕様説明書." (2020).

[9] Tapan kumar Das, and Manas Ranjan Mishra. "A Study on Challenges and Opportunities in Master Data Management." Int J Database Manage Syst 3.2 (2011): 129-39.

[10] 武田理宏, 真鍋史朗, and 松村泰志. "電子カルテデータ二次利用の現状と課題." 生体医工学 55.4 (2017): 151-158.

[11] Eisenberg, Andrew. "New standard for stored procedures in SQL." ACM SIGMOD Record 25.4 (1996): 81-88.

[12] Gray, Jim, et al. "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals." Data mining and knowledge discovery 1.1 (1997): 29-53.

[13] Melton, Jim, and Alan R. Simon. SQL: 1999: understanding relational language components. Elsevier, 2001.

[14] Boncz, Peter A., Marcin Zukowski, and Niels Nes. "MonetDB/X100: Hyper-Pipelining Query Execution." Cidr. Vol. 5. 2005.

[15] Manegold, Stefan, Peter A. Boncz, and Martin L. Kersten. "Optimizing database architecture for the new bottleneck: memory access." The VLDB journal 9.3 (2000): 231-246.

[16] Stonebraker, Mike, et al. "C-store: a column-oriented DBMS." Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker. 2018. 491-518.

[17] Patel, Jignesh M., Michael J. Carey, and Mary K. Vernon. "Accurate modeling of the hybrid hash join algorithm." Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems. 1994.

[18] Chen, Shimin, et al. "Improving hash join performance through prefetching." ACM Transactions on Database Systems (TODS) 32.3 (2007): 17-es.

[19] Ramesh, Sukriti, Odysseas Papapetrou, and Wolf Siberski. "Optimizing distributed joins with bloom filters." International Conference on Distributed Computing and Internet Technology. Springer, Berlin, Heidelberg, 2008.

[20] Barber, Ronald, et al. "Memory-efficient hash joins." Proceedings of the VLDB Endowment 8.4 (2014): 353-364

[21] Thi-To-Quyen, T. R. A. N., et al. "Improving Hamming distance-based fuzzy join in MapReduce using Bloom Filters." 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2018.

[22] Bloom, Burton H. "Space/time trade-offs in hash coding with allowable errors." Communications of the ACM 13.7 (1970): 422-426.

[23] Michael, Loizos, et al. "Improving distributed join efficiency with extended bloom filter operations." 21st International Conference on Advanced Information Networking and Applications (AINA'07). IEEE, 2007.

[24] Dawei Jiang, et al. "Cohort Query Processing." Proceedings of the VLDB Endowment, 2016.

[25] TPC. http://tpc.org/default5.asp （参照 2021-03-13）.