# 製造ビッグデータの検索における DB サーバへのインパクト推定

浪岡　保男† 　山形 康一† 　伊藤 竜一‡ 　鬼塚 真‡

久米本　幸平‡† 　服部 雅一‡‡ 　福田　雅允†

†(株)東芝　生産技術センター 〒235-0017 横浜市磯子区新磯子町 33

‡大阪大学大学院 情報科学研究科 〒565-0871 大阪府吹田市山田丘 1-5

‡†東芝デバイス&ストレージ(株)　IT 推進部 〒235-8522 横浜市磯子区新杉田町 8

‡‡東芝デジタルソリューションズ(株)　ソフトウェアシステム技術開発センター
〒183-8512 東京都府中市片町 3-22

E-mail: †{yasuo.namioka, koichi2.yamagata, masamitsu1.fukuda}@toshiba.co.jp,
‡{ito.ryuichi, onizuka}@ist.osaka-u.ac.jp,
‡†kohei.kumemoto@toshiba.co.jp, ‡‡masakazu.hattori@toshiba.co.jp

**あらまし**　製造現場から収集される製造データは，スマート・マニュファクチャリングなどとも関連し，年々，大規模化が進んでいる．データ規模は一つの工場でも日々億単位のレコード(数 TB)が集められ，製品に応じた保持期間分のデータを蓄積して全体で PB 級となる．この DB システムの検索処理が時々刻々と処理される中で，図らずも大量データ検索となるクエリを発行してシステムパフォーマンス低下を引き起こすことがある．この予防策として，クエリによる DB サーバへのインパクトの事前推定法が求められており，属性間の依存関係を非自己回帰モデルで学習し，推論時に与えられたクエリに応じてカーディナリティを推定する手法の導入を検討した．本報告では，この推定法を実用規模のデータに適用するためのデータサンプリング法とモデル分割法について報告する．大規模な DB サーバに対するクエリによるインパクトの推定に十分な精度が得られることを確認した．

**キーワード**　製造ビッグデータ，カーディナリティ推定，非自己回帰モデル，学習モデル分割，データサンプリング法

## 1. はじめに

スマート・マニュファクチュアリングやカーボンニュートラルなどの様々な取り組みが，製造現場においても進められている．これらの取り組みの土台として，AI や IoT などを駆使して，製造現場などにおける生産過程を網羅的にとらえるデータ収集が行われ，蓄積，分析が行われている．これら製造データの規模は，一つの工場においても，数 TB 程度の容量となっている．検査工程などからは，工程によっては日々数億や数千億のレコードが収集されている．製品にもよるものの，製品を顧客に納品した後でも，製造データの保持義務が契約されており，数年~十数年のデータが保持されている．このためデータの総量は PB 級となり，まさに製造ビッグデータとなっている．

製造ビッグデータの蓄積には，複数台の DB サーバからなる分散型のデータベースが採用されることが多い．分散型のデータベースでは，複数台の DB サーバに分割されてデータが蓄積されるとともに，様々な並列・並行処理により高速化が図られている．また，収集されたばかりのデータから過年度のデータまでをシームレスに検索することができる DB 構成とすることも可能である．

一方で，分散して蓄積されたデータは膨大であり，データ収集や分析などの処理において，適切な絞り込みを行わないと DB サーバの CPU，メモリ，ストレージ I/O などの計算リソースを一気に消費し，DB システム全体のシステムパフォーマンスを低下させてしまう危険がある．製造関連のシステムは，公開 DB ではないため，1 日あたり数十万アクセス程度であることが多いものの，影響は生産エンジニア，生産管理，工程品質管理，品質保証など多岐にわたることから，分析の自由度を保ちながら，システムパフォーマンス低下要因を避ける仕組みを備えることが求められている．

そこで，本研究では，クエリによる DB サーバへのインパクト，つまり，サーバリソースの必要量を事前に推定し，DB サーバへのインパクトが大きい場合には，ユーザやアプリケーション開発者に警告できる仕組みを検討している．サーバリソース必要量の推定には，ワークロードから回帰木を学習する手法[1]も提案されているが，本検討では，カーディナリティ推定法[2-13,23,24]，特に，属性間の依存関係を非自己回帰モデルで学習し，推論時に与えられたクエリに応じ精度高くカーディナリティを推定する手法[23,24]に着目した．この推定法を実用規模の製造ビッグデータに適用するためデータサンプリング法，モデル分割法，推定法について検討し，評価システムを構築して評価を行

った．本稿では，これらの方法とその評価結果について報告する．なお，インパクト推定対象の DBMS としては，GridDB[25]を想定した．

## 2. 関連研究

カーディナリティ推定法とこれらの方法を評価するためのデータセットとベンチマークについて取り上げる．

### 2.1. カーディナリティ推定法

カーディナリティ推定はデータベースのクエリオプティマイザの性能向上を主目的として長く研究されているが[2-12]，実際のデータベースシステムでは $10^4$-$10^8$×といったオーダーで推論のエラーが発生している．この結果として正確なカーディナリティを利用した場合と比較してクエリ応答性能で 100×以上性能低下することがあると知られている[13]．従来の手法では各々属性ごとに独立したヒストグラムを統計情報として管理しており，単一属性に対する選択演算のカーディナリティ推定は正確度が高いことが知られている．しかしながら，複数の属性に跨る選択演算や結合演算に関しては，各属性が独立であるという仮定が実際のデータと乖離しており，大きな推論のエラーとして表面化する課題があった．近年では属性間の相関関係を捉えられず推定エラーが発生してしまうという課題を解決すべく，機械学習を用いたカーディナリティ推定手法が提案されている[4-12]．これらは 2 つのアプローチに大別される．1 つはワークロードを学習することでカーディナリティ推定を行う手法である[6-8]．ワークロードが既知である場合や未知の場合でもランダムに生成されたワークロードを利用することで従来の手法と比較して高い性能を達成している．しかしながら,性能がワークロードに強く依存するため，未知のクエリに弱い・生成したワークロードのラベル取得コストが高く学習に時間がかかる・データベースが更新によって変化すると再学習が必要となるといった課題がある．

もう 1 つのアプローチとしてデータを学習するものが挙げられる[9-12,23,24]．このアプローチは更に，自己回帰モデル[14,15] や Sum-Product Networks [12, 18]を用いる方法と，非自己回帰モデル[19,20]を用いる方法[23,24]がある．

自己回帰モデル等を用いる方法では，学習時にデータの分布を捉え推論時に述語を適用することで従来の手法と比較して高い性能を達成している．また，データのみを学習に利用することから，事前にワークロードを必要としないという長所も挙げられる．しかしながら，大規模データの分布をカーディナリティ推定に利用可能な形で学習する必要があり，その安定性が課題となっている．例えば自己回帰モデルを利用した手法であれば，学習時に固定されてしまう推論可能な属性の順序によって Q-Error [13] で $10^2$－$10^3$× 上下する程度に性能が不安定であることが報告されている[8]．また，学習・推論コストの高さや扱える順序に制限があることから，多数の属性を扱うカーディナリティ推定への応用は行われていない．

非自己回帰モデルを用いる方法[23,24]では，非自己回帰モデルの密度推定としての性質を用いてデータを学習することで，属性ごとの分布だけでなく全リレーションの全属性間の相関関係を捉えたカーディナリティ推定を行う．学習時に推論できる属性の順序が固定されてしまう自己回帰モデルとは異なり，非自己回帰モデルは全属性を並列に扱うため任意の順序で推論できるという特徴がある．

### 2.2. データセットとベンチマーク

カーディナリティ推定の評価に用いられるデータセットとベンチマークとして DMV ベンチマーク[21]と，Join Order Benchmark [13](JOB-light [6])2 つのベンチマークを取り上げる．DMV ベンチマークはニューヨークの乗り物に関するデータセットを対象に動作する人工的に生成された 2000 個のクエリから成る．これらのクエリには 4～10 個の等号条件と範囲条件が含まれる．DMV データセットは単一リレーションであるため結合はない．Join Order Benchmark [13](JOB-light [6]) は,俳優，映像エンターテイメントやビデオゲーム等に関するデータセットである IMDb [22] を対象に動作する 70 個のクエリから成る．複数の等号条件と範囲条件があり，多数リレーションの自然結合も含む．ワークロードが事前に固定されている環境を想定して，IMDb データセットのうち，JOB-light に必要なリレーション・属性のみに絞り込んだデータセット(IMDb-JOB-light)も利用されている．各データセットの統計的な概要は表 1 に示す．

**表 1 データセット概要**

| データセット | リレーション数 | 行数 | 属性数 |
|---|---|---|---|
| DMV [21] | 1 | 11.6M | 11 |
| IMDb [22] | 16 | 4～36M | 2～17 |
| IMDb-JOB-light [6, 22] | 6 | 4～36M | 2～4 |

## 3. 製造ビッグデータの収集と活用

前提となる製造ビッグデータの収集と活用の流れと製造データの特徴について述べる．

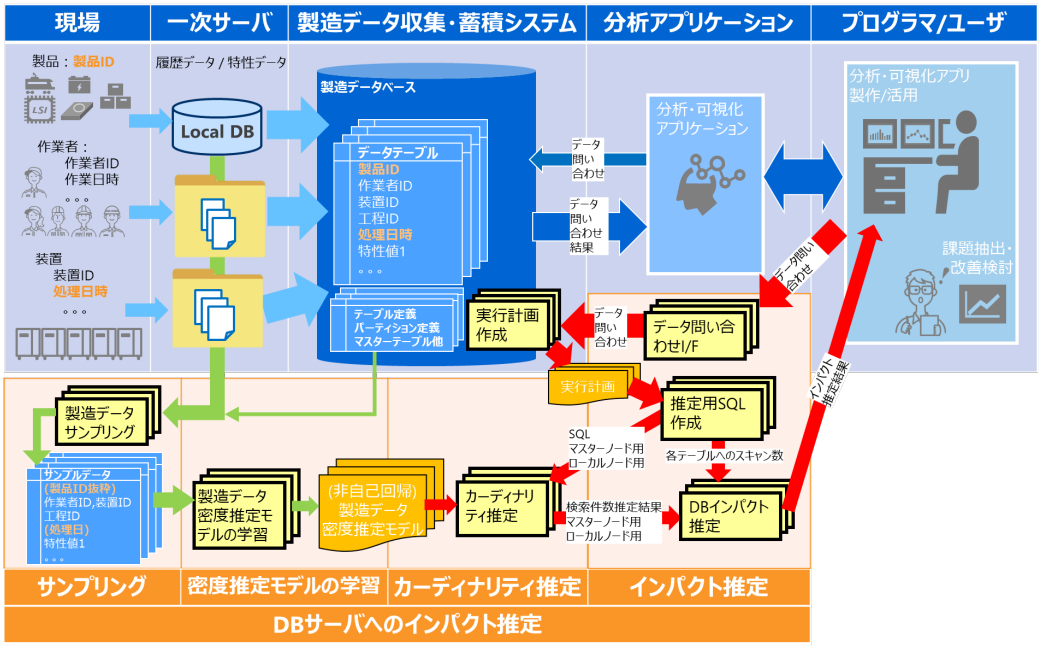


図 1 製造ビッグデータの収集活用の流れと DB サーバへのインパクト推定

### 3.1. 製造ビッグデータの収集活用の流れ

製造ビッグデータの収集活用の流れを図 1 上部に示す．製造現場には，製品，作業者，装置などがある．製造実行システムにより，製品には製品 ID が付与されている．作業者には作業者 ID が付与され，製造装置には，装置 ID が付与されている．作業実績や装置による処理実績には各々作業日時や処理日時が付与される．こうしたキー属性に対して，作業や処理の出来栄えや製品の特性などのデータが紐づけられる．

一次サーバには，製造実行システムにより製造現場にて収集された製造履歴や特性データが保存される．製造履歴や特性データは，製造現場ローカルな DB に登録される場合もあれば，装置ログなどとしてファイルとして保存される場合もある．これらのデータは，製造データベースに登録するため，一旦，登録用フォーマットに変換後に製造データベースに登録処理が行われる．

製造データベースは，1,000 テーブルを超す各種テーブルから構成されており，一次サーバからの製造履歴や特性データの他に，製品，工程などに関するマスターデータや，製造データベース自身のスキーマ情報などが記録されている．日々数 TB のデータが登録され，PB 級のデータが蓄積される．レコード数では数十兆レコードに達する．製造データベースは，これらのデータに対して行われる日々の十数万回のデータ収集，検索，集計処理に対応する．

### 3.2. 製造データの特徴

製造データベースに登録される製造ビッグデータには次のような特徴がある．

- 時々刻々と新しいデータが追記される
- 様々な製品が同時に流品する
- 製品 ID，作業者 ID，装置 ID 等の各種 ID や処理日時が付与されることで，データ集合間で共通部分(インターセクション)を無くしている
- ID を適切に選ぶことで，全数を集計しなくても母集団(個数：m)の傾向をつかむことができる
- ID や処理日時には欠損がありうる
- 製造工程が済々と運営されている場合は，先行工程の製品がほぼ次工程に流品するため，歩留り等の情報から前後の工程の流品数，つまり，データ数が凡そ推定できる
- 分析時には，日単位などで期間を指定することが多い
- データの到着にバラつきがあるので当日到着したデータの中には，当日から相当日数前までの処理日時のデータを含む可能性がある

製造ビッグデータのカーディナリティ推定と，これに基づく，検索における DB サーバへのインパクト推定は，これらの特徴に配慮した方法となる．

## 4. 製造 DB サーバへのインパクト推定法

分散 DB サーバへのインパクトを定義し，このインパクトの推定方針を述べる．更に，インパクト推定において用いるカーディナリティ推定法について述べる．

### 4.1. 分散型製造 DB サーバへのインパクトの定義と推定方針

分散 DB サーバへのインパクト，つまり，サーバリソースの必要量について，本研究では，サーバメモリの必要量に着目する．サーバメモリについて，図 2 GridDB におけるメモリ使用イメージを用いて述べる．GridDB はインメモリ DB ではないものの，大容量メモリを前提にメモリとストレージを併用してデータを格納する．そして，メモリ-ストレージ間のアクセス回数の削減，バッファ処理やリカバリ処理等の処理軽量化，データ処理時のロックフリー化を行うことで従来の RDB よりも高いパフォーマンスを得ている．GridDB のクラスタを構成する DB サーバのメモリは，

① データ用ストアメモリ，
② レコードフィルタ結果格納メモリ，
③ レコード結合集約結果格納メモリ

から構成される．上述のように，このメモリはサーバの物理メモリだけを指しているのではなくストレージを含めた記憶領域を指している．

クライアントからの検索要求に対応するために必要なデータは，ストレージから①データ用ストアメモリに吸い上げられる．吸い上げられる範囲は，データスキャン対象となるテーブル・パーティション数に依存する．

②レコードフィルタ結果格納メモリには，検索要求に関連する各々のテーブルに対して使用される索引やテーブル・パーティションにより絞り込まれたスキャン対象のデータが更に抽出される．このデータの中からクエリの全ての条件を満たすレコードが抽出されて，マスターノードに送られる．

マスターノードの③レコード結合集約結果格納メモリに集結したデータについて，結合や統計処理などの処理が行われて，最終的なクエリ結果としてクライアントに返される．マスターノードとしての処理は，DB クラスタの負荷状況により割り当てられ，マスターノードとしての処理は，適宜分散され負荷の平準化が行われている．以上のメモリ構成において，本研究では，分散型の製造 DB サーバへのインパクトを，次の 3 つに定義した．

① データ用ストアメモリの必要量
② レコードフィルタ結果格納メモリの必要量
③ レコード結合集約結果格納メモリの必要量

3 つの製造 DB サーバへのインパクトは，各々次のような方針で推定する．

① データ用ストアメモリの必要量:実行計画からデータスキャン対象となるテーブル・パーティション数を集計しこの合計から推定
② レコードフィルタ結果格納メモリの必要量:各々のテーブルに対して使用される索引やテーブル・パーティションにより絞り込まれたスキャン対象のデータ件数の合計から推定
③ レコード結合集約結果格納メモリの必要量:マスターノードに集まるレコード，つまり，各々のテーブルに対するクエリの全ての条件を満たすレコード数の合計から推定

これらのうち，②と③は，実行計画から得られる各々のデータテーブルに対して実行される単表毎の検索クエリについてカーディナリティ推定を適用する．推定方法は全く同じであるが，推定時に与える検索クエリの検索条件を調整することでどちらも推定することができる．

つまり，②については，スキャン対象となった各々のテーブルへのクエリについて，有効となる索引を検索条件としてカーディナリティ推定した結果(レコード数の合計)から推定することができる．③については，スキャン対象となった各々のテーブルへのクエリの全ての条件についてのカーディナリティ推定した結果(レコード数の合計)から推定することができる．

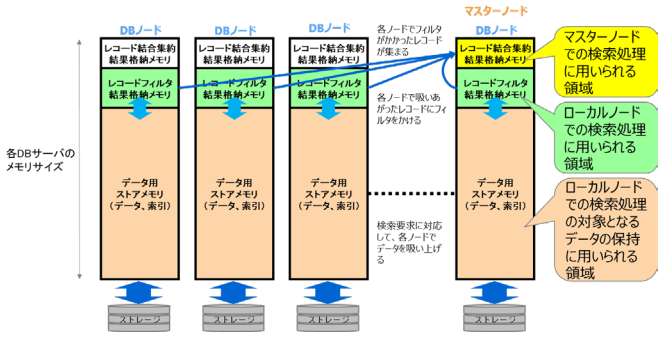


図 2 GridDB におけるメモリ使用イメージ

### 4.2. 製造 DB へのインパクト推定におけるカーディナリティ推定法

製造 DB へのインパクト推定法を図 1 の下側に示す．以下では，GridDB を用いて構築した DB システムを例に述べる．基本的には，クライアントからのクエリ要求により，②レコードフィルタ結果格納メモリや③レコード結合集約結果格納メモリに，検索対象となるデータテーブルから吸いあがるカーディナリティ(レコード数)によりメモリ必要量を推定する．

#### 4.2.1　データサンプリング

3.2 の製造データの特徴を活かし，以下の手順によりサンプリングを行って，学習用データを作成する．ここで，時刻カラムを日などの期間で丸めるとユニーク値の数を減らし学習モデルのサイズ削減に役立つ．また，サンプリングにおける m はロットサイズなどにより決定する．n は ID の欠損を想定して歩留りなどを参考に決定する．

**製造データサンプリング手順**

S:1 推定対象テーブル全てに以下の手順を行う
S:2　カーディナリティ推定に有効とするカラムを設定(基本は数値カラム以外)
S:3　サンプリング期間設定(当日 1 日などの期間)
S:4　サンプリング数設定(n/m)(m 個の中から n 個選ぶ)
S:5　サンプリング用カラム設定(製品 ID など)
S:6　サンプリング用カラムの値に対するサンプリング条件を設定(ex.製品 ID の下 2 桁を n 選ぶ)
S:7　丸める時刻カラムと丸め方の設定(日でまるめるなどで非数値化のようなことを行う)
S:8　推定対象テーブルに登録される Local DB やファイルから，サンプリング期間やサンプリング条件にてデータを抽出，また，抽出件数(num)も取得
S:9　抽出されたデータに対して，設定されたサンプリング用カラム，時刻カラムを丸める処理を実施
S:10 検索結果件数の推定時に DB 問い合わせの検索条件の中からチェック対象とするカラムのユニーク値を抽出
S:11 推定対象テーブル毎にサンプリングデータを出力

#### 4.2.2 密度推定モデルの学習

密度推定モデルの学習では，所定の期間に各々のテーブルに登録されたデータ毎に 4.2.1 のサンプリング法によって抽出されたサンプリングデータを用いて新たに密度推定モデル(非自己回帰モデル[35,36])を学習・作成する．学習手順を次に示す．

**密度推定モデルの学習手順**

L:1 全推定対象テーブルについて以下の処理を行う
L:2　推定対象テーブルに対して新たに作成する密度推定モデルを登録
L:3　カーディナリティ推定時に有効とするカラムを設定
L:4　新しい密度推定モデルがカバーする期間を設定
L:5　サンプリング数 (n/m)を設定
L:6　新しい密度推定モデルがカバーする期間に該当するサンプリングデータ(学習データ)読込
L:7　カーディナリティ推定時に有効とするカラムのユニーク値を抽出
L:8　サンプリングデータの件数(抽出したときの件数(num)を設定
L:9　非自己回帰モデルによる密度推定などの学習を実行，新しい密度推定モデルを出力

ここで，本研究では，非自己回帰モデルとして構造が平易で速度面を重視した多層パーセプトロン(MLP)(図 3)によるモデルを用いる．また，非自己回帰モデルによる密度推定の学習[23,24]は次のように行われる．学習はデータセット全体またはデータセットからランダムサンプリングしたタプル単位で行われる．リレーションからタプルを取り出し，マスクする属性を選択した後に，マスクしたものを入力，マスクしていないものをターゲットとする．そしてマスクした各属性 $X \in \boldsymbol{X}$ で，出力された確率分布 $\hat{P}_X$ とターゲットとなる確率分布 $P_X$ の交差エントロピーを算出し，その総和をマスク数|X|でスケーリングしたものをロス $\mathcal{L}$ として学習に利用する(数式 1)．タプルごとにマスクをランダムな数，ランダムな属性に選択することで，自己回帰モデルとは異なり，任意の属性を条件とした任意の属性の確率分布を推論可能なモデルとなる．

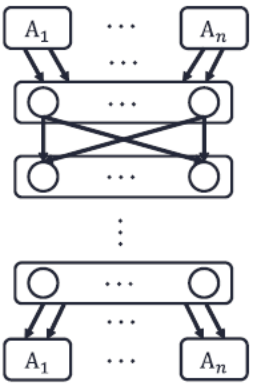


**図 3 MLP による非自己回帰モデル**

$$\boldsymbol{X} = \{A \in \boldsymbol{A} \mid A \ is \ masked\}$$

$$\mathcal{L} = -\frac{1}{|\boldsymbol{X}|} \sum_{X \in \boldsymbol{X}} \sum_{x \in X} P_X(x) \log \hat{P}_X(x) \quad \textbf{(数式 1)}$$

#### 4.2.3 カーディナリティ推定

カーディナリティ推定手順を次に示す．4.2.2 の学習にて作成された推定モデルを用いてカーディナリティ推定を行う方法では，②レコードフィルタ結果格納メモリの必要量，③レコード結合集約結果格納メモリの必要量を推定するために，②については，クライアントからのクエリに関連するテーブルに設定された索引に該当するカラムについての条件のみを使ったカーデ

ィナリティ推定を行う．このため，カーディナリティ推定は，②用に E:8，③用に E:7 にて各々同時確率を算出する．

また，サンプリングに用いたカラムが条件に含まれる場合とそうでない場合とでカーディナリティ推定が異なるため各々E:11，E:12 にて算出している．

なお，DB サーバへのインパクト判定では，E:11，E:12 にて算出されたカーディナリティ(レコード数)を各テーブルに対して予め設定しておいた閾値と比較して DB サーバへのインパクトの深刻度を判定する．

**カーディナリティ推定手順**

E:1 クライアントからのクエリを取得し，これに対する実行計画を取得

E:2 実行計画の末端(初段)の単一テーブルへのクエリを抽出

E:3 単一テーブルへのクエリのリストを作成

E:4 上のリストが空になるまで以下を実行

E:5 単一テーブルへのクエリを取得

E:6 単一テーブルへのクエリで指定されるテーブルについての密度推定モデルを(多数)選択

E:7 単一テーブルへのクエリにある検索条件に含まれるカーディナリティ推定で有効としているカラムに対する条件に基づいて，密度推定モデル(多数)により，条件に合致する行の同時確率を算出しそれらを合計

E:8 単一テーブルへのクエリにある検索条件に含まれるカーディナリティ推定で有効としているカラムに対する条件のうち，推定対象のテーブルの索引に合致する条件のみの条件に基づいて，密度推定モデル(多数)により，条件に合致する行の同時確率を算出しそれらを合計

E:9 E:7,E:8 の同時確率と各々のクエリ条件についてについて以下の計算を行い各々のカーディナリティを推定

E:10 if 検索条件がサンプリング用カラムを含？

E:11 then 条件に合致する行の同時確率×サンプリングデータ抽出数(num)÷n にてカーディナリティ推定

E:12 else 条件に合致する行の同時確率×サンプリングデータ抽出数(num)×m÷n にてカーディナリティ推定

ここで，前節の学習で得られたモデルを利用してカーディナリティ推定の基本的な手法[35,36]では，カーディナリティは述語を考慮した同時確率と総タプル数|R|の積と等価であることから，クエリ条件からなる条件付き確率からカーディナリティを推定している．

## 5. 評価実験

本研究のサンプリング法と推定モデルの分割法を用いても推定精度が保たれることを示すため，実際の工場の 1 か月分のデータを用いた評価実験を行った．以下では，データセット，実証システム，評価指標，実験結果の順に述べる．

### 5.1. データセットと実験方法

データセットとしては，50 品種以上が同時に流品している製品の製造工程のデータを用いた．このデータは，複数の検査工程の工程履歴が網羅的に記録されている履歴データと，1 つの検査工程から取得される製品の特性値のデータ(表 2)により構成してた．カーディナリティ研究で多く使用されているデータ(表 1)の 100 倍を超すレコード数のリレーションを含んでいる．また，実験方法としては，次のように行った．

- 製造ＤＢへのデータ登録日別にサンプリングデータを作成
- このデータに基づいて推定モデルも日別に作成し学習
- クエリに対する推定結果として，全ての推定モデルで推定し，この結果から得られるカーディナリティ推定結果を合計
- 同様のクエリを実際の製造 DB にて検索し検索件数を比較

この評価実験では，113 個のクエリに対して比較した．クエリは，歩留り集計等で分母の算出に通常使用される品種を特定して日別のデータ件数を集計するものと，特性値の統計処理の中で通常使用される製品 ID を指定したデータ件数を集計するものを模したものを使用した．

**表 2 評価に用いたデータセット概要**

| データセット | リレーション数 | 元データの行数 | サンプリング後の行数 | 属性数 |
|---|---|---|---|---|
| ある製品の検査工程を網羅した履歴データ1か月分 | 1 | 5M | 1M | 20 |
| 一つの検査工程の特性データ1か月分 | 1 | 13,000M | 100M | 10 |

### 5.2. 実証システム

図 1 の下側に示した DB サーバへのインパクト推定の仕組みは，次のようなスペックのサーバを用いて実証システムとして構築した．

CPU: Xeon Gold 6238R(28 コア, 2.2GHz)x2
GPU：nVidia RTX 3090 (24GB), 主メモリ：768GB,
OS：Ubuntu 20.04 (上の Docker を利用)
言語：Python3
Web アプリケーションフレームワーク:Django4

### 5.3. 評価指標

推定性能の評価指標には，真のカーディナリティを Cardinality，推定されたカーディナリティを $\hat{Cardinality}$ として(数式 2)で表される Q-Error [13]を用いる．これは推定されたカーディナリティが真の値から何倍離れているかを表す無次元の値であり，常に 1 以上で小さい方が優れていることを示す．

$$\text{Q-Error} := \frac{\max(\hat{Cardinality}, Cardinality)}{\min(\hat{Cardinality}, Cardinality)} \quad \textbf{(数式 2)}$$

### 5.4. 実験結果

サンプリング，学習，推定に要した処理時間を表 3 にまとめる．サンプリング時間と学習時間はデータ量の多い特性値の処理により多くの時間を要していた．一方で，推定時間はモデルサイズが比較的小さくなる特性値の方が若干短時間で処理されていた．サンプリングと学習の時間は 1 日分のデータに対する処理時間である．また，推定時間には，クライアントからのクエリに対する実行計画作成，①検索対象となるテーブル・パーティション数の集計(データ用ストアメモリ必要量に直結)，②索引カラムのみの条件でのカーディナリティ推定(レコードフィルタ結果格納メモリの必要量に直結)，③クエリ条件全てを用いたカーディナリティ推定(レコード結合集約結果格納メモリの必要量に直結)が含まれる．

**表 3 処理時間**

| | 履歴データ | 特性値データ |
|---|---|---|
| サンプリング時間 | 10 秒～35 秒 | 100 秒～350 秒 |
| 学習時間 | 100 秒～170 秒 | 5 千秒～1 万 5 千秒 |
| 推定時間 | 14 秒 | 9 秒 |

また，113 個のクエリについて，推定値と実システムでの検索件数との比較結果を表 4 にまとめた．この精度は，DMV[22]を用いた評価実験[36]の精度からそれほど乖離しておらず，サンプリングやモデル分割を行っても精度が維持できているといえる．更に，DB サーバへのインパクトを推定する目的においては，検索件数の桁数(1 万件，10 万件，100 万件，1,000 万円，1 億件など)で閾値を設定することが想定されるため，Q-Error が 10 を超えないことが望ましい．特に，特性値データは 1 か月で 100 億件を超えるデータとなることから最大でも Q-Error が 5 未満であることは大変良好な結果であるといえる．履歴データはデータの母数が比較的少ないことから，95%th で 10 未満であればこちらも良好な結果が得られたといえる．

また，特性値データの工程処理日別の検索件数を推定した結果と，実システムでの検索結果とを工程処理日日別に比較した結果を図 4 にまとめた．データサンプリングと密度推定モデルの学習は，データ登録日ベースで実行されているが，データ登録日は工程処理日に対して，製造工程の運用上の都合により 1 時間から数日の遅れが発生する．このため，同じ工程処理日のデータに関する推定を複数のモデルで実行するが，このように実行することで，サンプリングや学習の区切りとして用いた時刻とは異なる工程処理日をベースにしても，実数の変動に推定値がしっかりと追随できているといえる．

**表 4 評価実験における Q-Error**

| | | | Median | 90%th | 95%th | 99%th | Max |
|---|---|---|---|---|---|---|---|
| 履歴 | 製品 A | 日別集計 | 1.47 | 2.89 | 3.66 | 8.28 | 12.77 |
| | 製品 B | 日別集計 | 1.66 | 3.61 | 8.51 | 11.65 | 18.90 |
| 特性値 | 製品 A | 日別集計 | 1.03 | 1.07 | 1.12 | 1.53 | 1.73 |
| | 製品 A | 製品 ID 指定 | 1.29 | 1.52 | 1.53 | 1.57 | 4.81 |

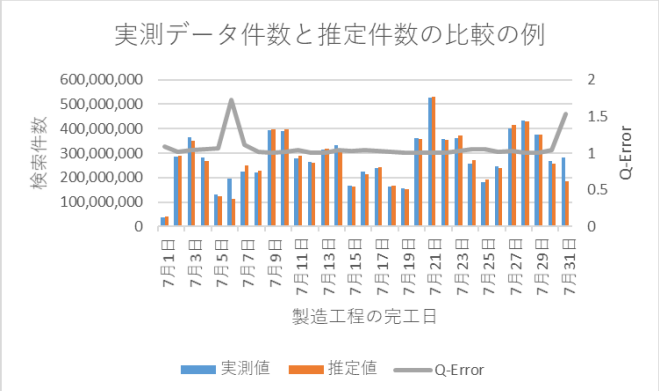


**図 4 実システム上の検索件数と推定件数との処理日別比較**

## 6. まとめ

本報告では，製造ビッグデータの検索における DB サーバへのインパクト推定について述べた．この中で，実際に稼働中の大規模システムのデータに対して，非自己回帰モデルで学習し，推論時に与えられたクエリに応じて高精度にカーディナリティ推定する手法を紹介した．本報告のデータサンプリング法，モデル分割法，推定法を用いることで，PB 級の大規模な DB システムに対するインパクトの推定に十分な精度が得られることを確認した．

謝辞　本研究の一部では，(一)日本データベース学会のプロダクト提供型アカデミック支援プログラムによる GridDB の貸与の支援をいただきました．また，カーディナリティ推定の密度推定モデルの学習ならびに推定の基礎的方法は，国立研究開発法人新エネルギー・産業技術総合開発機構（ＮＥＤＯ）の委託業務（JPNP16007）の結果得られたものです．

## 参　考　文　献

[1] Li, J., König, A.C., Narasayya, V., and Chaudhuri, S.: Robust Estimation of Resource Consumption for SQL Queries using Statistical Techniques, Proceedings of the VLDB Endowment, Vol. 5, No. 11, PP1555-1566 (2012).

[2] Poosala, V., Ioannidis, Y. E., Haas, P. J. and Shekita,E. J.: Improved Histograms for Selectivity Estimation of Range Predicates, Proceedings of the ICDE (1996).

[3] Chow, C. and Liu, C.: Approximating discrete probability distributions with dependence trees, IEEE Transactions on Information Theory, Vol. 14, No. 3, pp.462–467 (1968).

[4] Heimel, M., Kiefer, M. and Markl, V.: Self-Tuning, GPU-Accelerated Kernel Density Models for Multidimensional Selectivity Estimation, Proceedings of the SIGMOD, pp. 1477–1492 (2015).

[5] Kiefer, M., Heimel, M., Bres, S. and Markl, V.: Estimating join selectivities using bandwidth-optimized kernel density models, Proceedings of the VLDB Endowment, Vol. 10, No. 13, pp. 2085–2096 (2017).

[6] Kipf, A., Kipf, T., Radke, B., Leis, V., Boncz, P. and Kemper, A.: Learned Cardinalities: Estimating Correlated Joins with Deep Learning, Proceedings of the CIDR (2019).

[7] Kipf, A., Vorona, D., M¨uller, J., Kipf, T., Radke, B., Leis, V., Boncz, P., Neumann, T. and Kemper, A.: Estimating Cardinalities with Deep Sketches, Proceedings of the ICDE, pp. 1937–1940 (2019).

[8] Woltmann, L., Hartmann, C., Thiele, M., Habich, D. and Lehner, W.: Cardinality estimation with local deep learning models, Proceedings of the Workshop on aiDM, pp. 1–8 (2019).

[9] Yang, Z., Liang, E., Kamsetty, A., Wu, C., Duan, Y., Chen, X., Abbeel, P., Hellerstein, J. M., Krishnan, S. and Stoica, I.: Deep Unsupervised Cardinality Estimation, Proceedings of the VLDB Endowment, Vol. 13, No. 3, pp. 279–292 (2019).

[10] Yang, Z., Kamsetty, A., Luan, S., Liang, E., Duan, Y., Chen, X. and Stoica, I.: NeuroCard: One Cardinality Estimator for All Tables, Proceedings of the VLDB Endowment, Vol. 14, No. 1, pp. 61–73 (2020).

[11] Hilprecht, B., Schmidt, A., Kulessa, M., Molina, A., Kersting, K. and Binnig, C.: DeepDB: Learn from Data, not from Queries!, Proceedings of the VLDB Endowment, Vol. 13, No. 7, pp. 992–1005 (2019).

[12] Zhu, R., Wu, Z., Han, Y., Zeng, K., Pfadler, A., Qian, Z., Zhou, J. and Cui, B.: FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation, Proceedings of the VLDB Endowment, Vol. 14, No. 9, pp. 1489–1502 (2021).

[13] Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A. and Neumann, T.: How good are query optimizers, really?, Proceedings of the VLDB Endowment, Vol. 9, No. 3, pp. 204–215 (2015).

[14] Germain, M., Gregor, K., Murray, I. and Larochelle, H.: MADE: Masked Autoencoder for Distribution Estimation, Proceedings of the ICML (2015).

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I.: Attention Is All You Need, Proceedings of the NeurIPS (2017).

[16] Uria, B., Murray, I. and Larochelle, H.: A Deep and Tractable Density Estimator, Proceedings of the ICML, p. 467––475 (2014).

[17] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. and Le, Q. V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding, arXiv preprint (2019).

[18] Poon, H. and Domingos, P.: Sum-Product Networks: A New Deep Architecture, Proceedings of the AAAI (2017).

[19] Gu, J., Bradbury, J., Xiong, C., Li, V. O. K. and Socher, R.: Non-Autoregressive Neural Machine Translation, Proceedings of the ICLR (2018).

[20] Ghazvininejad, M., Levy, O., Liu, Y. and Zettlemoyer, L.: Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Conference on EMNLP (2019).

[21] State of New York: Vehicle, Snowmobile, and BoatRegistrations, https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations (2019).

[22] IMDb: Internet Movie Data Base, https://www.imdb.com (1990).

[23] 伊藤竜一，佐々木勇和，肖川，鬼塚真：Non-Autoregressive モデルによる高速で安定したカーディナリティ推定，日本データベース学会 DEIM Forum 2021 E11-2, Mar., 2021.

[24] 伊藤竜一，佐々木勇和，肖川，鬼塚真：非自己回帰モデルによる高速で安定したカーディナリティ推定，情報処理学会論文誌データベース（TOD）Vol.15(3), pp.36-49, Oct., 2022.

[25] GridDB: https://www.global.toshiba/jp/products-solutions/ai-iot/griddb.html