NNI Technologies

ビッグアーカイブ*1のRAGを実現する

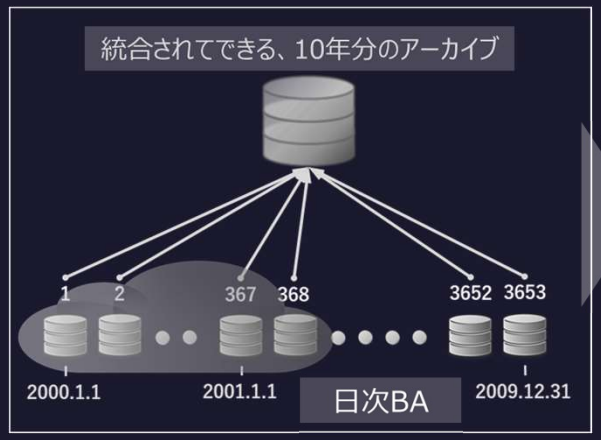
Peta Book

*1. ビッグアーカイブ（BA）とは、蓄積され続け膨大な規模になった
表形式データのアーカイブです。

2025.02.01 rev.17

ビッグアーカイブ（BA）*1 の利用: 3つの課題

課題1. 膨大な統合処理時間



課題2. 遅い検索
（統合してもインデックスが無い）

課題3. 保管・転送時間
（有意義に使えたデータを保管・転送できない）

*1. ビッグアーカイブ（BA）とは、蓄積され続け膨大な規模になった表形式データのアーカイブです。

BA-RAGのアクセスのリアルタイム性： 3つの課題

BA-RAGを実現するためには、BAへのリアルタイムアクセスが不可欠ですが、以下の3つの課題が特に障害となっています。

課題1. 膨大な統合処理時間

通常、BAは多数のファイルに分割されており、そのままでは利用が困難です。しかし、それらを統合しようとすると膨大な統合処理時間がかかります。

課題2. 遅い検索

統合後のBAには、一般的にインデックスが付与されていません[*1]。そのため、事前に想定できない多様なクエリに対応するにはフルスキャンが必要となります。これでは、どれほど高速な計算環境を用いてもリアルタイムの応答は困難です。しかし、もし適切なインデックスがあれば、100億レコードの検索でもミリ秒単位で実行可能です。

課題3. 保管・転送時間

統合後のBA、その検索やソートの結果としてできるBAは、一時的もしくは長期的な保管が必要になります。また、共有や別の処理との連携のために転送も必要になります。これらにも時間がかかります。

これらの課題を解決するのが「**写像表形式データ（MTD[*2]）**」です。

*1. BAの1カラムに対するインデックス構築時間ですら膨大です。ましてや多数のカラムにインデックスを付加し、分析に役立つようにすることは非現実的です。

*2. Mapped Table formatted Data の略です。

例えば10年間の日次アーカイブを参照する場合、3,653個の日次アーカイブを読み出し、統合する必要があります。これには膨大な時間を要します。

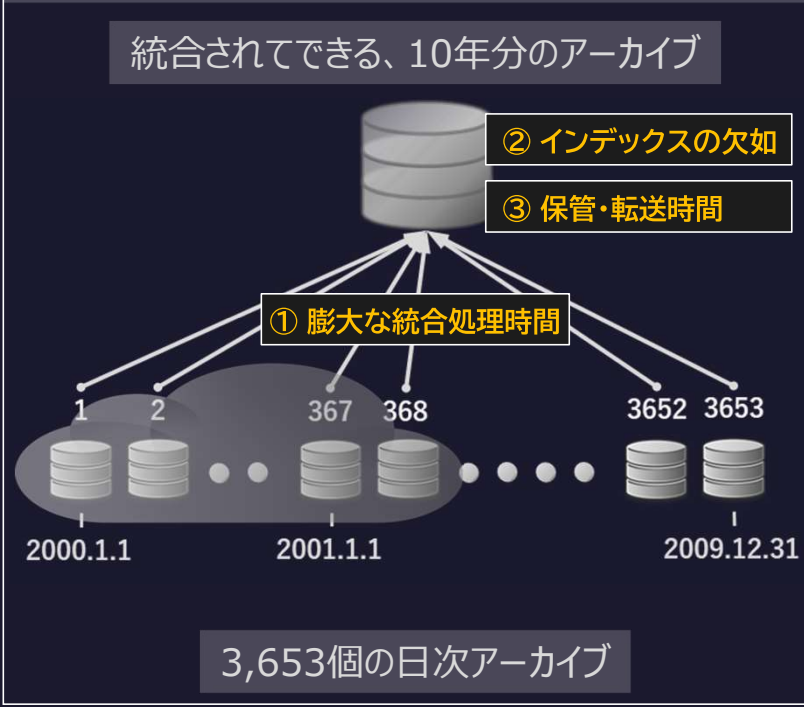


図-1. 10年分の日次データの利用上の課題

3つの課題を解決する「写像表形式データ(MTD*1)」

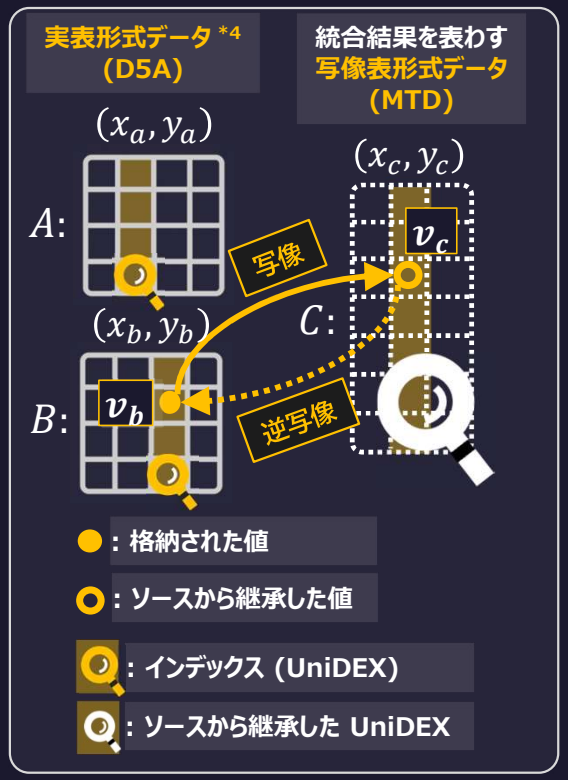


図2. 実および写像表形式データ間を結ぶ **写像**、**逆写像**

1. MTD*1は、ソースからの写像で定義され値を持たない。

2. MTD上の値は、必要になったときに逆写像を辿ってソースから取得。
   ⇒ 統合時間＝0 *2
   ⇒ 代数的表現であるためコンパクトで保管・転送が容易 *2

3. MTD上の、全てのカラム、全てのカラムの組合せ、全ての部分集合に有効なインデックス：UniDEX *3 が自動的に付く。
   ⇒ リアルタイムに検索ができる

*1. Mapped Table-formatted Data のアクロニム
*2. 付録2（パフォーマンスレポート）参照
*3. Universal Data Exploration Index のアクロニム
*4. 値とUniDEXを保持。クラウド等のリモートに存在していても良い。

3つの課題を解決する「写像表形式データ（MTD）」

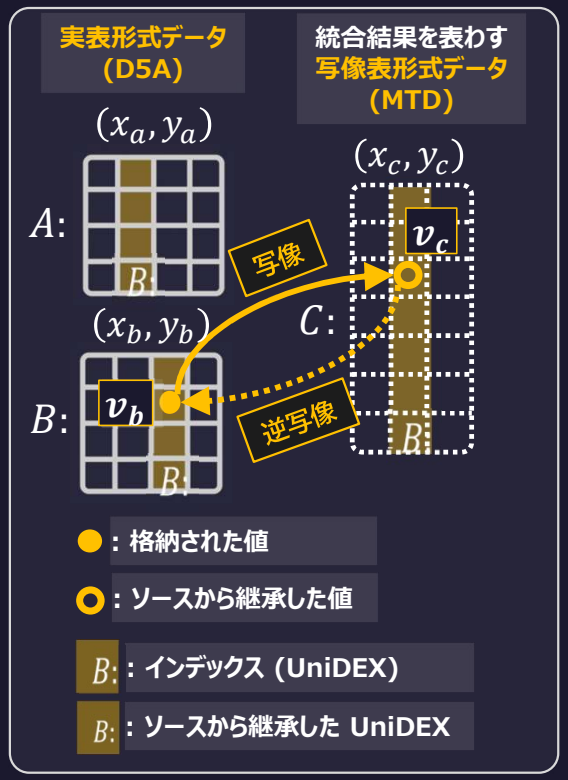


図-2. 実および写像表形式データ間を結ぶ写像、逆写像

統合処理時間、保管・転送時間 の解決

リアルタイム性の課題のうち、**統合処理時間、保管・転送時間**は、必要になったところだけを必要になったタイミングで値を転送する仕組みで解決できます。これを「**遅延転送**」と呼ぶことにします。

遅延転送を実現するためには、ソースとデストのアドレス (x, y) を対応させる必要があります。ソースからデストには写像を、デストからソースには逆写像を使います。（左図参照）

写像は多階層にできますが、それらのオリジンには値を保持する実表形式データ(D5A)が必要です。

実表形式データ（D5A[*2]）は、**値** を保持し、次ページで紹介する **UniDEX**[*3] というインデックスを備えています。このD5Aからの写像を受け入れるのが写像表形式データです。

写像表形式データ（MTD[*1]）とは、ソースからの写像で定義され、**値を保持しない**表形式データです。このソースとはD5Aもしくは他のMTDです。

MTDには以下のメリットがあり、前記の課題1～3を全て解決します。

1. BAの統合がリアルタイムにできる
   遅延転送により統合時間がかからない
2. MTD生成と同時に UniDEX が備わる
   MTDは**D5Aから UniDEX を継承できる**ため
3. 保管・転送が容易
   MTDは値を保持しないのでコンパクトであり、保管・転送が容易に行える

※ UniDEX の詳細は付録1で説明しています。
また、そのパフォーマンスは付録2で報告しています。

*1. Mapped Table formatted Data のアクロニムです。
*2. 第5世代アーカイブ から付けられた名前です。
*3. Universal Data Exploration indeX の短縮形です。

Peta Book の構造

どのBAにも共通して必要とされるアプリ

例：グリッド表示、グラフ描画、簡易SQL

個々のBAに対応したアプリ

例：BOM展開、類似レコード検索（後述）

エンジンは API 経由でアクセスされる

BA はMTD(写像表形式データ) で構成されているのでコンパクト

D5A(実表形式データ) は Peta Book の外側にある

BigQuery VS Peta Book

	BigQuery	Peta Book
仕組み	分散並列コンピューティング	フルインデックスな分散テーブルファイル機構
動作環境	クラウド上（限定的な実行環境）	ファイルシステム上（幅広い実行環境）
可能な処理	幅広い	インデックスで加速可能な処理とその応用に限定
コスト	高い（多数のCPUを使う）	低い（SSD上に格納するだけ）
レスポンス	多くの処理が何秒、それ以上	多くの処理が1秒以下、しばしばミリ秒単位
データ更新	可能	追記のみに限定
データ統合	長時間	リアルタイム
データの保管・転送	長時間	リアルタイム

2025年初頭に提供する Peta Book のUI

※1. グラフ表示機能は2025初頭では搭載されない

Peta Book が対話型AIにもたらす新たな可能性

AIはPBを通じてさまざまなBAに同時にアクセスし、個々のBAおよびBA間の連携から多様なデータを抽出し、それらを基にして回答が行えるようになります。

右の囲いは、対話型AIが数兆レコードのコネクティッドカーデータ（CCD）および製造トレーサビリティデータへのリアルタイムアクセスを通じて、迅速かつ的確に情報を引き出し、価値のある回答を生成する近未来のシナリオを描いています。

この例のように、Peta Bookは対話型AIの活動領域を飛躍的に拡大することができます。

囲い-1

鈴木さんの Xモデル3 のバッテリーが異常です。

顧客サービス担当者:

鈴木さんのEVのバッテリーが時々突発的に低電圧になります。

AI:

鈴木さんは2年前に Xモデル3を購入され、ほぼ毎日8時間利用されています。鈴木さんのコネクティッドカーデータ(CCD)は 2,102,498 レコードで突発的な低電圧は 13 回発生しています。搭載するバッテリーは P303-A です。
現在のXの全CCDは 6,307,223,102,386 レコードであり、P303-Aに関するレコードは 2,111,923,100,222 レコードです。しかし、類似の事象は、Y工場、Z工場で生産された 112 台に、全部で 981 回しか発生しなかった極めて稀なものです。推定される原因は3つです。

- 熱暴走: 35.1%
- サイクル寿命劣化: 42.8%
- 電圧不均衡: 22.1%

まずXバッテリー技術部: 03-XXXX-XXXX に連絡して下さい。次にバッテリーの状態診断を行った上で、ファームウェアのアップデートをして下さい。

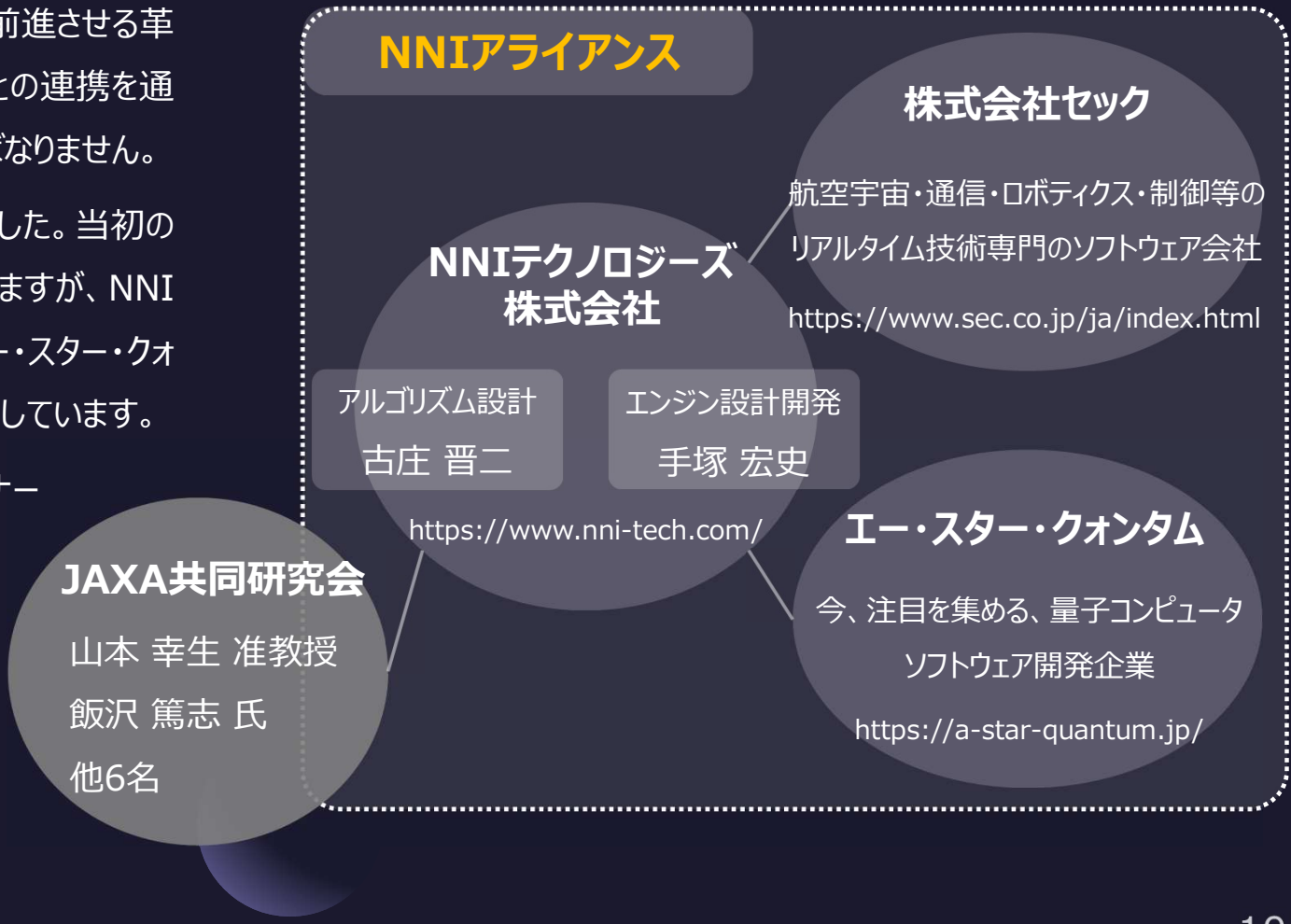
Peta Book を支えるNNIアライアンスとパートナー

Peta Bookは次世代のデータ活用を大きく前進させる革新的なソリューションであるため、多くの企業との連携を通じたグローバル展開を是非とも推進しなければなりません。

その中心となるNNIアライアンスが結成されました。当初のNNIアライアンスは、スモールスタートではありますが、NNIテクノロジーズを中心に、株式会社セック、エー・スター・クォンタムなど、先端技術をリードする企業が参加しています。

また、JAXA共同研究会もその重要なパートナーとなっています。

これらにより、今まさに、Peta Bookが最初の一歩を踏み出そうとしています。

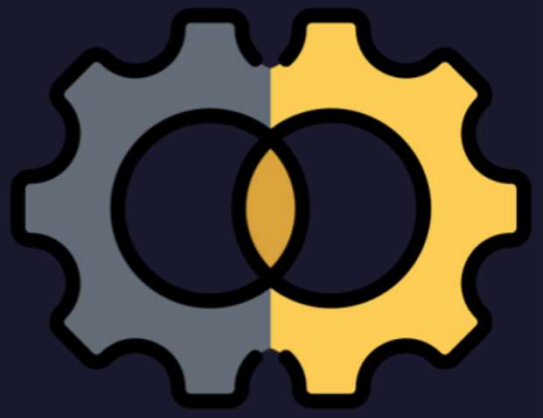


付 録

付録１．用語と仕組み

付録２．Peta Book のパフォーマンスレポート

付録１.用語と仕組み



1. 用語集

実表形式データ とは、値 と UniDEX (後述) を格納した表形式データ。

D5A とは、実表形式データを格納するためのファイルフォーマット。または実表形式データの略語。

MTD*1（写像表形式データ） とは、Mapped Table formatted Data のアクロニムで、値を持たず、ソース表形式データからの写像で定義された表形式データ。値を保持しないので、数兆レコードを表現しているときもコンパクト。

D5AVU とは、MTDを格納するためのファイルフォーマット。写像の定義を保持するだけなのでコンパクトである。

UniDEX とは、あらゆるカラム、カラムの組合せ、部分集合に対して有効なインデックスである。実表形式データ上では構築済み、MTD上では実表形式データから継承され自動的に出現する。

BA とは、蓄積され続け膨大な規模になる表形式データのアーカイブ。

PB とは、MTDで表わされた「BA」、BAにアクセスするための「エンジン」、および「アプリ」を1つにまとめ、人やAIが簡単にBAを入手し利用することを可能にしたもの。

NNI とは、Natural Number Index のアクロニムで、表形式データの全てのカラムを自然数化することで、その検索・集計・ソート・マッチング・集合演算などの幅広い処理を高速化するアプローチを言う。

NNI代数 とは、MTDやMTD上のUniDEXを実現するために使われる数学的枠組み。

*1. Natural Number Index に由来

2. 自然数インデックス（NNI）の仕組み

自然数インデックス(NNI[*1]) とは、表形式データの**全てのカラムを自然数化**することで、その検索・集計・ソート・マッチング・集合演算などの幅広い処理を高速化するアプローチを言います。

1. カラムの自然数化とは

下図の SVL (Sorted Value List) はオリジナルのカラムの中に出現する値をユニーク・昇順に並べたものです。また、図中の NNC (Natural Numbered Column) は、オリジナルのカラムの値を、SVL中の格納位置で置き換えたものです。

SVLを除けば、NNCをはじめとするいくつかの自然数配列だけでカラムを表現しなおすことができます。これがカラムの自然数化です。

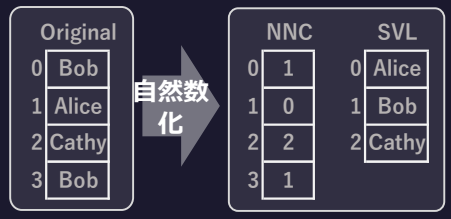
以降、SVL、NNC、…の各配列を**成分**と呼びます。

2. 幅広い処理の高速化とは

SVLによりカラム全体で何種類の値があるか、最大/最小値は何か、が直ちに分かります。それら以外の成分でもそれぞれ分かることがあります。

これらの成分は例えば A[B[i]] のように結合が可能です。成分をさまざまに組み合わせることで直ちに結果が出せる処理（例えばある種の検索、ある種の集計、ある種のソートなど）があります。

それらを何ステップか組み合わせることは可能で、そこまで含めると、幅広い処理が記述できます。これらの処理は本質的に必要なことしか行っていないため高速です。



3. NNI代数とは

前記のNNIのアプローチは実表形式データ内で一歩一歩積み上げられてきました。ところが、MTDや MTD上のUniDEXまで実現しようとすると、数学的枠組みが必要になります。その要請に応えるのがNNI代数です。

例えば、図-2に示されるMTD上の座標 (x_c, y_c) 上の値 v_c は、逆写像で求まる座標 (x_b, y_b) 上の値 v_b であり、代数的な値です。より複雑な写像や多段階の写像の場合、NNI代数無しに v_c を求めることは困難です。

このようなNNI代数は、これまでNNI代数無しに開発されてきたアルゴリズムもシンプルに記述可能にし、それらを大変分かりやすいものにしています。

*1. Natural Number Index に由来

3. UniDEX の仕組み

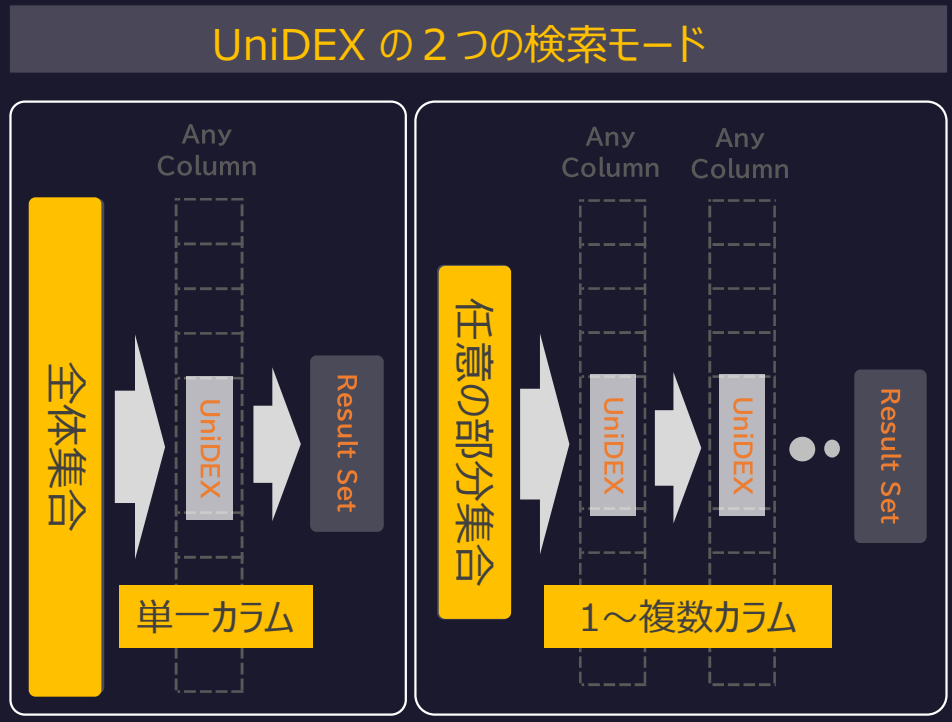
UniDEXは、D5A上に存在し、MTD上でもD5Aから継承して使えるインデックスです。

UniDEXは、全体集合や部分集合から、任意の1～複数のカラムに条件を設定して絞り込むことで検索を行います。

1. **高速モード**
   任意の1カラムに条件を設定して、全体集合から絞り込みます。このモードは、単純で高速な検索が必要な場合に適しています。
2. **高自由度モード**[*1]
   任意の1～複数のカラムに条件を設定して、任意の部分集合からさらに絞り込みます。このモードは、柔軟性が求められる複雑な検索に適しています。

これらの2つの検索モードを適切に使い分ける、あるいはミックスして使うことで、様々な検索要求に迅速かつ柔軟に応答することが可能になります。

*1. このモードの拡張として「類似ベクトル検索」が実装されます。



高速モード
任意の1カラムに対し、全体集合からの検索を高速に行えます

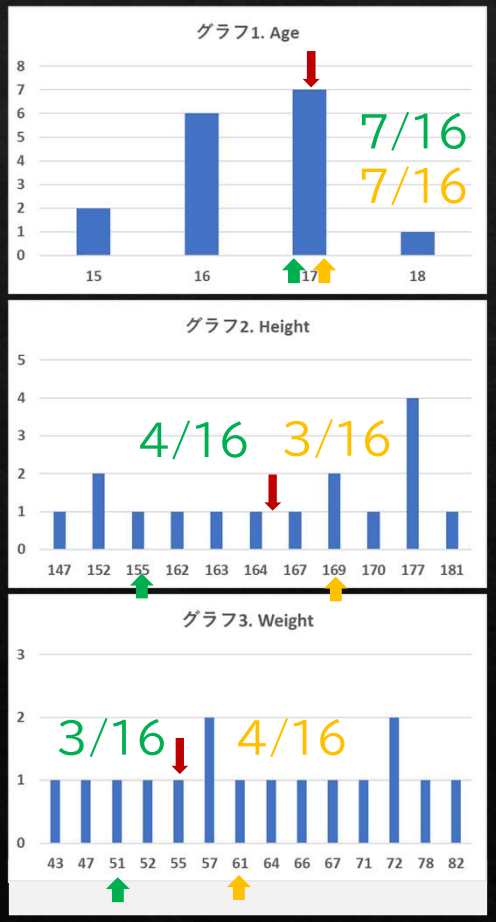
高自由度モード
任意の1～多数カラムに対し、任意の部分集合からの自由度の高い検索を行えます

類似レコード検索

（自己情報量に基づく）類似レコード検索

表1.例とするデータ

	Age	Height	Weight
0	15	162	72
1	15	163	47
2	16	164	67
3	16	181	57
4	16	177	72
5	16	152	64
6	16	177	66
7	16	152	57
8	17	169	78
9	17	155	51
10	17	177	82
11	17	169	61
12	17	177	52
13	17	170	71
14	17	147	43
15	18	167	55



検索目標とするカラムのセットを任意に選び、それらのカラムのセットに対する目標値（クエリベクトル）を与え、クエリベクトルに近い値が格納されているレコードを探す検索を類似レコード検索と呼ぶ。

クエリベクトルとレコードとの間の類似度は、クエリベクトル上の各要素とレコード上の各要素の間の情報量の総和で測られる。その情報量は、カラムの分布上で、クエリベクトル上の値とレコード上の値を両端とする分布の区間上に出現する値の個数の、全レコード数に対する割合を生起確率と見なし、その生起確率の対数で与えられる。

例えば、17歳、165cm、55kgをクエリベクトルとすると、以下のような検索ができる。

レコード: 9 では -log(7/16) -log(4/16) -log(3/16)= 5.61 bit、
レコード: 11 では -log(7/16) -log(3/16) -log(4/16)= 5.61 bit、

従って、レコード 9 と 11 のいずれも、クエリベクトルと 5.61 bit離れていると言える。
（なお、この2つのレコード以上にクエリベクトルに近いレコードは表1中には存在しない。）

付録 2.
Peta Book の パフォーマンスレポート

1. Peta Book の総合パフォーマンス
2. 配送トラックデータでの UniDEX のパフォーマンス測定(1/3)
3. 配送トラックデータでの UniDEX のパフォーマンス測定(2/3)
4. 配送トラックデータでの UniDEX のパフォーマンス測定(3/3)
5. パフォーマンスのまとめ

1. Peta Book の総合パフォーマンス

3つの高速化機構

MTD は統合時間、保管・転送、検索の3つの高速化を達成しました。
統合および保管・転送の高速化は MTD そのものの仕組みに由来し、検索の高速化は MTD 上のあらゆる場所に対して有効なインデックスである、UniDEX に由来します。一方、データ取得は1つ1つのセル毎に逆写像を辿る仕組み上、レイテンシーはともかく、スループットは低めになります。
ここでは、3つの高速化機構のパフォーマンスを調べていきます。

#1 - 統合のパフォーマンス

ソースデータ群を統合して作られる MTD は、ソースからの写像で作られています。そのため、MTDが行う統合は、以下になります。

1. 写像の定義を読み込む。
2. ソースとのコネクションを確立。

デモムービー***1**では、1000個のソースを統合し、31.5兆レコードにするのに 0.3秒かかりました。

#2 - 保管・転送のパフォーマンス

MTD はソースから MTD への写像の定義でできています。そのため、どんなに巨大な表形式データを表現しているときでも極めてコンパクトになります。コンパクトであるため、保管・転送がリアルタイムに行えます。
デモムービー***1**では 756PB の表形式データを僅か 512KB で表現しています。この場合、Saveや転送は 0.1秒程度と見積もることができます。

#3 - UniDEXのパフォーマンス

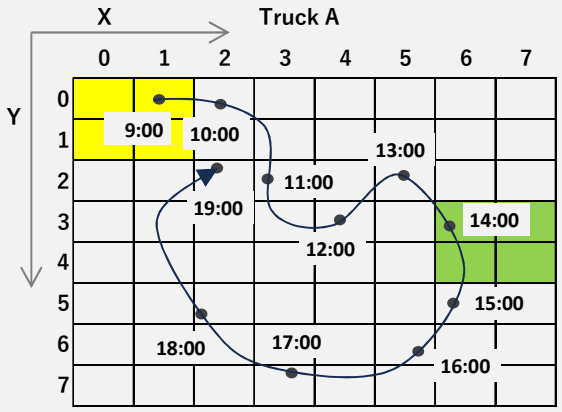
次項以降で解説します。

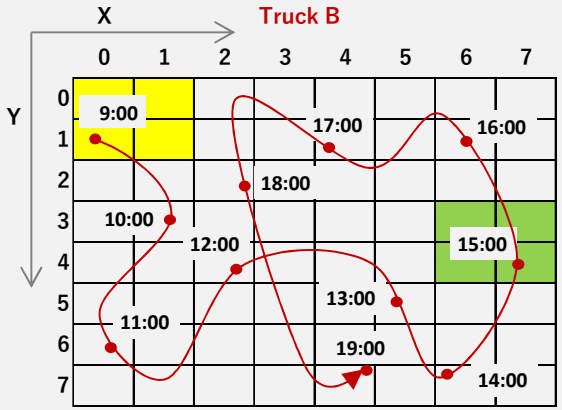
*1 https://youtu.be/wkH49DmSPSQ

2. 配送トラックデータでの UniDEX のパフォーマンス測定（1／3）

	Time		X	Y	kg
0	9:00	A	1	0	2000
1	9:00	B	0	1	1000
2	10:00	A	2	0	1700
3	10:00	B	1	3	800
4	11:00	A	3	2	1600
5	11:00	B	0	6	300
6	12:00	A	4	3	1200
7	12:00	B	2	4	1100
8	13:00	A	5	2	1100
9	13:00	B	5	5	900
10	14:00	A	6	3	800
11	14:00	B	6	7	800
12	15:00	A	6	5	700
13	15:00	B	7	4	500
14	16:00	A	5	6	600
15	16:00	B	6	1	400
16	17:00	A	3	7	400
17	17:00	B	4	1	300
18	18:00	A	2	5	250
19	18:00	B	2	2	200
20	19:00	A	2	2	0
21	19:00	B	4	7	0

10億レコード
13カラム





ベンチマークに用いたデータ

ー 10億レコードのトラック配送記録 ー
（プログラムで生成した疑似データ）

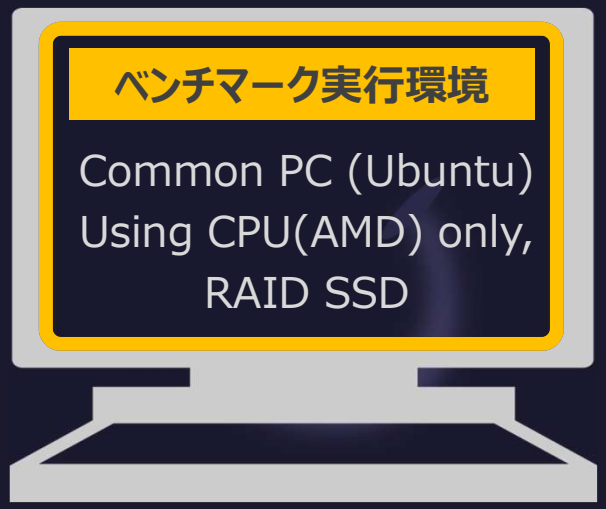
1. 日本全国に5000台の配送トラックを運用する大手運輸を想定。
2. 1分ごとに、各トラックから、緯度・経度・積載量など13カラムのデータを収集したとの想定。
3. 1年366日で総レコード数: **989,297,868**
4. 13カラム
   ①支店名, ②日付時刻, ③経度, ④緯度, ⑤トラックID,
   ⑥ドライバーID, ⑦助手人数, ⑧積載個数, ⑨積載個数率,
   ⑩積載重量kg, ⑪積載重量率, ⑫積載体積m3, ⑬積載体積率

このデータセットを用いて、次に UniDEX の実際のパフォーマンスを検証します

3. 配送トラックデータでの UniDEX のパフォーマンス測定（2／3）

Benchmark-1

高速モード



16通りの 高速モード のパフォーマンス

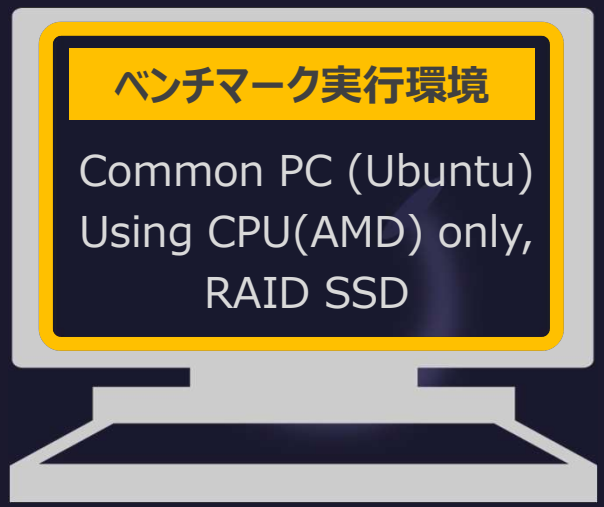
検索条件式の [A, B) は A≦ x ＜ B 、[A, B] は A≦ x ≦ B を表わす

測定#	カラム	検索条件	ヒット数	検索時間(msec)
#1	支店名	札幌市支店	19,779,180	0.62
#2	支店名	札幌市支店|千葉市支店|さいたま市支店|横浜市支店		
			118,712,212	0.14
#3	日付時刻	[2024-03-01,2024-06-01)	248,718,584	0.75
#4	日付時刻	[10:00:00,11:00:00)	109,800,000	5.49
#5	緯度	[30.00,41.00)	946,076,633	2.12
#6	緯度	[35.00,36.00)	284,216,014	1.84
#7	経度	[130.00,140.00]	777,949,877	1.95
#8	経度	[139.00,140.00]	177,995,757	1.39
#9	ドライバーID	[1000000,1030000]	435,283,772	0.22
#10	ドライバーID	[1030000,1040000]	217,679,924	0.08
#11	積載重量kg	[100.0,1300.0)	794,703,211	0.36
#12	積載重量kg	[10.0,20.0)	9,110,234	0.1
#13	積載個数	[50,150)	534,256,412	0.16
#14	積載個数	[100,110)	54,764,041	0.06
#15	助手人数	[0,2]	989,297,868	0.17
#16	助手人数	1	494,656,740	0.06

4. 配送トラックデータでの UniDEX のパフォーマンス測定（2／3）

Benchmark-2

高自由度モード



2通りの 高自由度モード のパフォーマンス

検索条件式の [A, B) は　A≦ x ＜ B 、[A, B] は　A≦ x ≦ B を表わす

測定#	カラム	検索条件	単純検索ヒット数
# 1	支店名	札幌市支店	19,779,180
	日付時刻	[08:00:00,09:00:00)	109,800,000
	緯度	[43.05,43.08)	1,544,843
	経度	[141.32,141.35)	2,014,679
	ドライバーID	[1010010,1010020)	1,979,120
	積載重量kg	[900.0,1100.0)	88,771,362
	積載個数	[80,130)	268,968,867
	助手人数	1	494,656,740
複合検索のヒット数:　351　実行時間(msec): 194.82			

測定#	カラム	検索条件	単純検索ヒット数
# 2	支店名	東京本店|千葉市支店|さいたま市支店|横浜市支店	
			118,712,212
	日付時刻	[08:00:00,09:00:00)	3,100,000
	緯度	[43.05,43.08)	211,029,583
	経度	[141.32,141.35)	64,082,133
	ドライバーID	[1010010,1010020)	118,712,212
	積載重量kg	[900.0,1100.0)	66,292,133
	積載個数	[80,130)	268,968,867
	助手人数	1	494,656,740
複合検索のヒット数:　303　実行時間(msec): 614.60			

5. パフォーマンスのまとめ

これまでのパフォーマンステスト結果を総括すると以下となります。

- 1000個のビッグアーカイブの統合が **1秒未満**、
- 1兆レコードを超えるデータの保管・転送が **1秒未満**、
- 10億レコードに対する、「高速モード」での検索が **1ミリ秒前後**、
- 10億レコードに対する、「高自由度モード」での検索が **1秒未満**、

以上のパフォーマンスの結果から、MTD と MTD上の UniDEX を用いると、ビッグアーカイブにリアルタイムにアクセスできることが分かります。

これにより、**ビッグアーカイブへのリアルタイムアクセス** と、その結果として **BA-RAGを実現**し、AIの活用領域の大幅な拡大が可能になることは明らかです。

サマリー：ビッグアーカイブ(BA)の RAG を実現する Peta Book

ビッグアーカイブ（BA）とは？

ビッグアーカイブ（BA）は、蓄積され続けることで巨大な規模になる表形式データのアーカイブです。

1. 統合BA作成の課題

BAは日次データのように多数に分割されていることが多く、かつ、使用目的が限定されていないことが普通です。このため、BAを効率よく利用するには、多数のソースとなるBAから必要なカラムを選び出したり、値を変換(例:単位の変換)したり、適切な場所にデータを配置し直したりして統合し、統合BAを作る必要があります。しかし、それには長い時間がかかります。

2. インデックスの課題

対話型AIがBAを利用する場合は、秒以下のレスポンスが求められます。それにはインデックスがなければ対応できません。しかし、既存のインデックスは構築に長い時間がかかる上、特定のカラムにしか有効ではありません。

3. 保管・転送の課題

作成した統合BA、およびその検索結果等は保管する必要があることがしばしばです。また、それらの共有や販売のためには転送の必要があります。しかし、いずれも長い時間がかかります。

MTD(写像表形式データ)とは？

MTDは値を保持せず、ソースからの写像で値が定義される表形式データで、前記の3つの課題を解決します。

1. 写像で定義され、値が必要になったときに取得する遅延転送の仕組みを持つため、BAのリアルタイム統合ができます。
2. MTD上にはインデックス：UniDEX が自動的に備わります。このUniDEX はMTD上のあらゆるカラムとカラムの組合せ、あらゆる部分集合に有効です。
3. MTDは極めてコンパクトであり、保管・転送が容易です。

Peta Book（PB)とは？

PBとは、MTDで表わされた「BA」、BAにアクセスするための「エンジン」、およびそのBAを使いこなすための「アプリ」を1つにまとめパッケージ化したものです。このようなPBにより、人やAIがBAを入手し利用することが簡単になります。

近未来には多種多様なPBが出版され、さまざまなBAを手軽に活用できるようになることでしょう。その結果、科学技術が発展し、社会・経済活動が効率化され、新たなサービスが生まれることでしょう。

Thank you

https://www.nni-tech.com/

contact@nni-tech.com