多次元データベース設計のための一提案

テクニカルコンピテンシー部
シニアITスペシャリスト

福島　祐宏

Yasuhiro　Fukushima
yasuhiro-fukushima@exa-corp.co.jp

OLAP分野で活用される多次元データベースは、事前集計した結果を格納するためさまざまな切り口でのデータ分析、検索を高速に行える反面、データ投入・事前集計処理に多大な時間を必要とし、システム構築・運用の妨げになる場合がある。

本稿では、多次元データベースの構造特性に注目し、

1）構成次元の疎/密定義を性能設計上の着眼点とすべきであること

2）構成次元の疎/密定義のために考案した技法の有効性

を実証結果をもとに解説する。

1. はじめに

蓄積されたデータをさまざまな視点から自由に切り口を変えて多次元的に分析するテクノロジーと概念を1993年、IBMサンノゼ研究所のエドガー・F・コッド（E.F.Codd）博士は12個のルールと共にOLAP（online analytical processing)として提唱した[1)]。企業内に蓄積されたデータを戦略的に活用する上でこの概念は極めて有効であり、この提唱以降、OLAP実現のツール（以下、OLAPツールと記す）がベンダー各社から提供され[2)]、企業内情報活用分野で数多く利用されてきている。

現在、各ベンダーから提供されているOLAPツールはROLAP系のものとMOLAP系のものに大別できる。

ROLAPではデータをリレーショナル・データベースに蓄積し、分析用のSQLを発行することによりデータの多次元的な分析が可能となり、分析の柔軟性に優れているが、１回の処理に多大なコストが必要になる。

MOLAPでは、データとその事前集計結果を「キューブ」という特殊な格納構造をもつ多次元データベース内に格納するため、多次元検索が高速にできる反面、キューブへのデータ投入や事前集計処理に多大なコストが必要になる。また、ROLAPがデータ格納用としてリレーショナル・データベース、クエリー言語としてSQLという標準の部品で構成されるのに比べ、MOLAPの「キューブ」とクエリーにはベンダー間での互換性はない。さらに、MOLAPで使用される多次元配列は、一般に（1）疎配列となり、(2)集約演算や検索の速度が配列の次元に依存する、という問題点がある[3)]。これらの問題点に対応するため、データ格納、圧縮技術などの学術的な研究も継続的に行われている。

本稿では、既存の商用MOLAP製品をいかにうまく活用するかという利用技術の観点から、多次元データベース設計・構築上の課題であるデータ投入・事前集計処理に焦点をあて、その最適化手法について提案する。まず、著名なベンダー製品を採用して実施した検証結果をもとに、実設計・構築においては、構成次元の疎/密定義がその性能を決定する重要な要素であることを示す。さらに、最適な疎/密定義を行うために、投入データがあらかじめリレーショナル・データベースに格納されていることを想定し、SQLによる事前シミュレーション手法を提案し、性能最適化のための指針を述べる。最後に、その指針の有効性を実データにて確認する。

2. 多次元データベースの設計・構築と課題

本章では、まず、多次元データベース設計・構築の概要と、課題であるデータ投入・事前集計処理について述べる。引き続きこれら課題の要因となる多次元データベースでのデータの物理格納構造について、検証に使用した製品をもとに解説する。なお、商用MOLAP製品の物理実装は、ベンダーに依存する部分も多々あるが、多次元データベースの基本構造に大きな差はないと考える。

2.1. 設計・構築のプロセスと課題

多次元データベースの論理構造、設計・構築プロセスの概要、課題について、図１に示す３次元キューブの例に基づき、以下解説する。

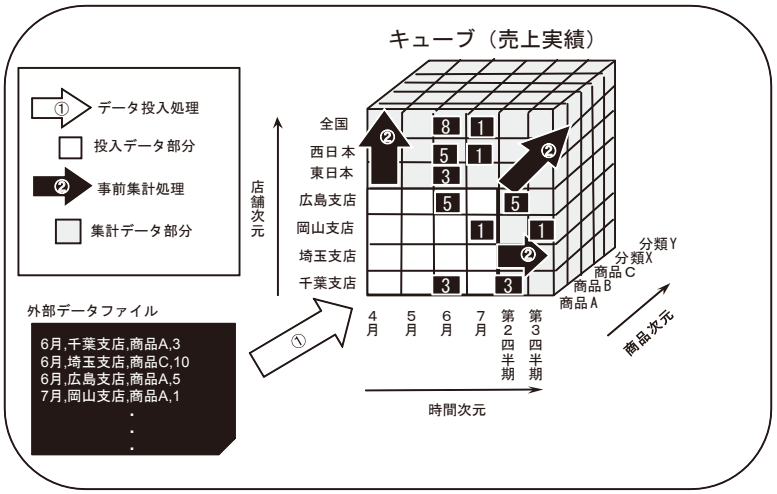


図１　多次元データベース

多次元データベースではデータを、「キューブ」と呼ばれる多次元の構造体に格納する。構築にあたっては、まず、「キューブ」を構成する次元軸を設計・定義しその枠組みを決定する。１つの次元軸はその次元を構成する基本項目とそこから派生させる集計項目により構成する。図１の例では、時間次元を構成する基本項目は「４月」、「５月」、「６月」、「７月」であり、「第２四半期」、「第３四半期」は集計項目である。また、店舗次元を構成する基本項目は「千葉支店」、「埼玉支店」、「岡山支店」、「広島支店」であり、「東日本」、「西日本」、「全国」は集計項目である。さらに、商品次元を構成する基本項目は、「商品A」、「商品B」、「商品C」、であり、「分類X」、「分類Y」は集計項目である。

次元軸の設計・定義を終了し、枠組みが完成すると、集計・分析の対象とする実績データを外部データファイルからキューブ内に投入する（図１ 矢印①）。

投入した実績データは、キューブ内のセルと呼ばれる記憶領域に格納される。図１中、キューブ内白枠の部分が、投入データを格納するセル領域となる。領域内で、どのセルに格納されるかは、投入する実績データにより決まる。実績データは、各次元を構成する基本項目の値と、これらに対応する実績項目の値で構成される。商品Aの６月の千葉支店での売上実績が３であった場合、実績データは“６月,千葉支店,商品A,３”となり、売上実績値3は時間次元の「６月」、店舗次元の「千葉支店」、商品次元の「商品A」で決定されるセルに格納される。データ投入を完了した時点でのキューブの状態は、図２に示すようなリレーショナル・データベース上のテーブルの状態と等価となる。

図１のキューブにおいて、投入データ部分の値が格納されたセルは、図２内実績テーブルの１レコードに相当する。また、図１のキューブを構成する各次元は、図２内各マスター・テーブルに相当する。

データ投入を完了すると、セル内に格納されたデータをもとに各次元軸上に定義した集計項目の積み上げ演算をキューブ上で行う（図１矢印②）。これを事前集計処理と呼ぶ。図３に店舗次元に対する事前集計処理の例を示す。

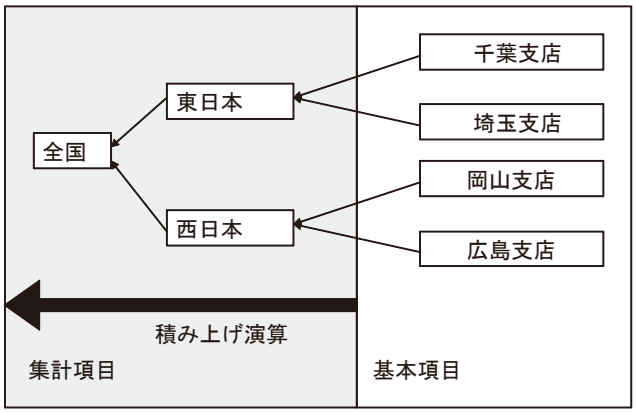


図３　事前集計処理（店舗次元）

この処理は、多次元データベースが提供する機構で行う。

事前集計処理の結果もキューブ内のセルに格納される。図１中、キューブ内網掛けの部分が、集計結果を格納するセル領域となる。店舗次元に対する集計処理結果を例にとると、商品Aの６月の東日本での売上実績は、千葉支店での実績３のみであったので、「商品A」-「６月」-「東日本」に対応するセルには集計結果として３が格納される。

商品ﾏｽﾀｰﾃｰﾌﾞﾙ

商品	商品名
A	商品A
B	商品B
C	商品C

実績ﾃｰﾌﾞﾙ

売上月	商品	店舗	売上実績
06	A	001	3
06	A	004	5
07	A	003	1

時間ﾏｽﾀｰﾃｰﾌﾞﾙ

売上月	名称
04	4月
05	5月
06	6月
07	7月

店舗ﾏｽﾀｰﾃｰﾌﾞﾙ

店舗	店舗名
001	千葉支店
002	埼玉支店
003	岡山支店
004	広島支店

図２　リレーショナル・データベース

また、商品Aの６月の西日本での売り上げは広島支店の実績５のみであったので、「商品A」-「６月」-「西日本」に対応するセルには集計結果として５が格納される。商品Aの６月の全国での売上実績は、東日本での３と西日本での５を集計した８が、「商品A」-「６月」-「全国」に対応するセルに格納される。

こうして、キューブが完成する。

このように、多次元データベースでは、対象データをあらかじめ事前集計し、その結果をすべてキューブ内に保持しているため、高速な検索・分析が可能となる。

企業内に蓄積された膨大なデータを有効活用するには、次元数、次元を構成する項目数、集計項目数などある程度の規模をもったキューブが必要である。しかし、キューブが大規模化した場合には、その構築のためのデータ投入・事前集計には多大な処理時間が必要となる。実システムにおいては、データ投入・事前集計処理は通常、夜間のバッチ作業で行うことになる。定められたバッチ処理時間帯の中でいかに効率よく、これらの処理を終了させることができるかがMOLAP型のシステム導入、多次元データベース設計・構築の課題である。

この課題の一因に多次元データベースにおけるデータの物理格納構造がある。次節でその仕組みを解説する。

2.2. 物理格納構造

多次元データベース設計・構築にあたっては、キューブを構成する次元ごとに、「疎」「密」の定義・設定を行う。セルは「密」定義した次元の組み合わせで構成されるデータ・ブロックに格納される。「疎」定義した次元の組み合わせはこれらデータ・ブロックへのポインター（索引）となる。

図１の論理構造をもつ多次元データベースで、店舗次元を「疎」とし、商品次元、時間次元を「密」とした場合のデータ格納構造を図４に示す。

データ・ブロックに格納されるセルにはデータを保持するものと、データを保持しないものが生じる。データを保持しないセルも記憶域を必要とする。本稿では、実際にデータを保持しているセルを実データセル、データを保持していない空のセルを非データセルと記す。図４中、黒色のセルが実データセルであり、白色のセルは非データセルである。例えば、「千葉支店」-「５月」-「商品A」に対応する売上実績はないので、このセルは、非データセルである。

一般に、多次元データベースでは実データセルの存在は数%以内と言われており、大量に記憶領域を占めてしまう非データセルの発生を抑制できるよう、次元の「疎」「密」定義を適切に行う必要がある。設計者の感覚により誤った定義を行うとデータ投入・事前集計処理の際、予期せぬパフォーマンストラブルを招く。次章にてテストデータによる検証結果をもとにその重要性を述べ、疎/密次元定義最適化設計に必要な要件を考察する。

3. 疎/密次元定義が性能に及ぼす影響

集計結果をキューブ内に格納することにより､多次元データベースは高速な検索・分析を可能としている。しかし、

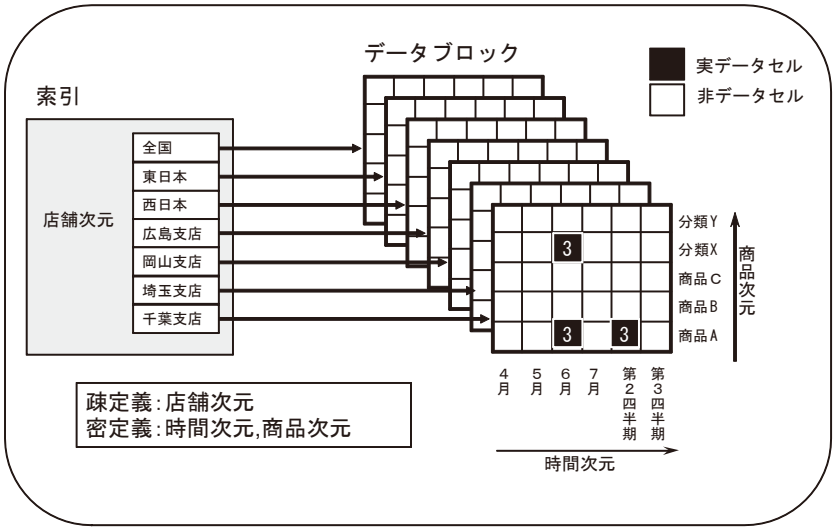


図４　物理格納構造

前章で述べた構造特性を考慮すると、構成次元の疎/密定義パターンによりデータ投入、事前集計に要する処理時間は大幅に異なることが予想できる。

本章では、疎/密次元定義パターンによるデータ投入、事前集計処理テストの内容と結果をもとにこの点を考察する。

3.1. テスト内容

3.1.1. テストデータ仕様

本テストで使用したデータはOLAP Council (http://www.olapcouncil.org)が提供する４次元のベンチマークテスト用データである。その仕様を表１に示す。

表１　テストデータ仕様

次元データ			
次元名称	基本項目数	集計項目数	項目数合計
Customer	900	100	1,000
Product	9,000	1,000	10,000
Channel	9	1	10
Time	24	10	34
実績データ			
データ密度(%)		データ件数	
0.1		1,239,300	
1.0		12,393,000	

実際には、これらのデータは、OLAP Councilが提供する専用ツールで生成する。その際、0.1%から10%の範囲で「データ密度」を指定する。ここで、「データ密度」とはキューブの投入データ部分が、「疎」になる度合いを表現する指標である。例えば図１で示した論理的な多次元キューブの投入データ部分がすべて実データセルで埋まっていればデータ密度は100%と考えればよい。

論理キューブの投入データ部分が、「疎」になるかどうかは、投入する実績データにより決まる。以後、本稿ではキューブを「疎」な状態にしてしまう投入データを“「疎」なデータ”と表現する場合もある。

実システムにおいては、多くの場合データ密度は数パーセント以内になると言われているが、社内データも含め、いくつかの実データを試算してみたところいずれも１%以内の密度となった。したがって、本テストでは「データ密度」0.1%と1.0%の2種類のデータを生成して使用した。

0.1%を指定した際、1,239,300件/78MB、1.0%の場合には、12,393,000件/780MBの実績データを得ることができる。

3.1.2. テスト環境

テストは当社社内検証用UNIX系サーバ(HP-UX 11,CPU:440MHz PA8500x2,MEMORY 2GB)を使用して行った。

3.1.3. テスト手順

疎/密次元定義が性能に与える影響、傾向を捉えるため、表２に示すように４個の構成次元のうち３個を「疎」とした場合(A)、２個を「疎」とした場合(B)、１個を「疎」とした場合(C)、３つの次元定義パターンそれぞれに対し、

多次元キューブ定義→実績データ投入→事前集計処理を繰り返し、データ投入、事前集計処理の処理時間を計測した。

表２　疎/密次元定義パターン

疎密パターン	次元名称			
	Time	Product	Customer	Channel
A	D	S	S	S
B	D	S	S	D
C	D	D	S	D

(注): D: 密 (Dense) , S: 疎 (Sparse)

3.2. テスト結果

データ投入、事前集計に必要とした処理時間を表３に示す。

表３　処理時間の差異

データ密度	疎密パターン	データ投入時間 (HH:MM:SS)	集計処理時間 (HH:MM:SS)
0.1%	A	0:03:42	0:18:57
	B	0:06:48	0:41:06
	C	1:07:59	10:20:12
1.0%	A	0:59:12	2:00:31
	B	0:55:24	8:57:42
	C	15:20:04	20:01:27

全く同一のデータを使用しているにもかかわらず、疎/密次元定義のパターンにより、データ投入処理時間、集計処理時間ともに大きく変動した。

データ密度0.1%の場合、投入データの集計処理は、最速

であったAの定義パターンでは約19分で終了したが、Cでは完了までにその30倍以上の10時間20分を必要とした。データ投入時間もAとCの間では20倍以上の相違が生じた。

データ密度1.0%の場合も、Aが最速ではあるが、Cとの間での処理時間比は、データ投入についてはA:C=1:15、集計処理についてはA:C=1:10と小さくなる。

この結果は、データが「疎」である場合、「密」とする次元を多く定義すると処理時間は相対的に大きくなることを示し、また、データが「疎」であるほど疎/密次元定義が性能設計上、重要であることを示している。

3.3. 考察

これら、疎/密次元定義による性能の変動は、図４で示した多次元データベースの格納構造に起因するものと考える。

以下、この格納構造に注目し、性能最適化のための方向性について考察する。

3.3.1. データ投入について

実績データは「密」定義した次元で構成されるデータ・ブロック内に実データセルとして格納されていく。しかし、投入データは通常はランダムな構造となる。したがって投入データを「疎」定義した次元の組み合わせに沿ってソートしておけば、各データ・ブロックに順次格納されることになり、安定したデータ投入性能を期待できる。

3.3.2. 事前集計処理について

集計処理時間の変動は、次元の組み合わせにより発生するデータ・ブロックの量、非データセルの量が大幅に異なることに起因する。表４にデータ投入完了時点、表５に事前集計処理完了時点でのキューブの状態を示す。表中、データセル数、非データセル数は製品提供のユーティリティー・コマンドで得た結果を掲載した。当然のことながら、データ投入完了時点ではデータセル数は投入データ件数と一致する(表４)。しかし、非データセル数は構成次元の疎密パターンにより大幅に異なる。条件B,Cのパターンでは発生した大量の非データセルの影響でデータベース容量が肥大化し、結果的に多大な集計処理時間が必要になったことを示している。

特に、投入データが「疎」である場合、非データセルの発生を抑制できるように、疎/密次元定義を行えれば、最適な事前集計処理性能を期待できる。

表４　データ投入完了時点でのキューブの状態

データ密度	疎密パターン	データ投入後　キューブの状態			
		非データセル数	データセル数	合計セル数	DBサイズ(bytes)
0.1%	A	1,239,300	1,239,300	2,478,600	17,383,424
	B	23,546,700	1,239,300	24,786,000	110,182,400
	C	1,739,560,700	1,239,300	1,740,800,000	885,080,064
1.0%	A	12,393,000	12,393,000	24,786,000	141,115,392
	B	212,068,200	12,393,000	224,461,200	956,383,232
	C	2,163,607,000	12,393,000	2,176,000,000	3,765,448,704

表５　集計処理完了時点でのキューブの状態

データ密度	疎密パターン	集計処理後　キューブの状態			
		非データセル数	データセル数	合計セル数	DBサイズ(bytes)
0.1%	A	7,883,316	21,898,100	29,781,416	290,537,472
	B	94,388,020	21,898,100	116,286,120	978,403,328
	C	1,963,701,900	21,898,100	1,985,600,000	3,975,688,192
1.0%	A	44,112,285	122,534,125	166,646,410	1,604,927,488
	B	388,382,515	122,534,125	510,916,640	4,288,774,144
	C	2,298,265,875	122,534,125	2,420,800,000	9,621,209,088

4. 性能最適化指針

前章でのテスト結果と考察をもとに、多次元データベースにおけるデータ投入、事前集計処理性能最適化の指針を本章で記述する。

4.1. データ投入性能最適化

前章と同じテスト条件にて、「疎」定義した次元の組み合わせに沿ってデータを事前ソート後、キューブに投入した結果を事前ソートなしで投入した結果との比較で図５、図６に示す。なお、事前ソートはOS付属のsortコマンドで実施した。

事前ソートそのものにはデータ密度1.0%の場合、最大100分程度の処理時間を必要とし、その効果も疎/密次元定義パターンにより異なったものとはなったが、特に「密」次元を多く含む場合には、投入データは、「疎」と定義した次元に沿って、あらかじめソートしておくことが望ましい。「密」と定義した次元を多く含むパターンCについては、データ密度0.1%で7倍(図５)、1.0%で15倍程度(図６)の性能向上を得た。

4.2. 事前集計処理性能最適化

前章の考察では、「非データセルの発生を抑制できるように」とその方向性までは示したが、どの次元を「密」とし、どの次元を「疎」とするかは、難題である。実際には非データセルの発生状況は疎/密次元定義を行った後、データを多次元データベースに投入してみないことにはわからない。データ投入以前に、疎/密次元パターンに応じたキューブ内の状況を予測できないものであろうか。

この課題解決にむけて、考案した手法を以下提案する。

本手法は非データセルの発生状況を簡易なSQLにて事前算出し、発生する非データセルが最小になるように次元の疎/密構成を行うことを導くものである。なお、投入データは、リレーショナル・データベース内に実績表として格納されていることを前提とする。

4.2.1. 事前シミュレーション手法

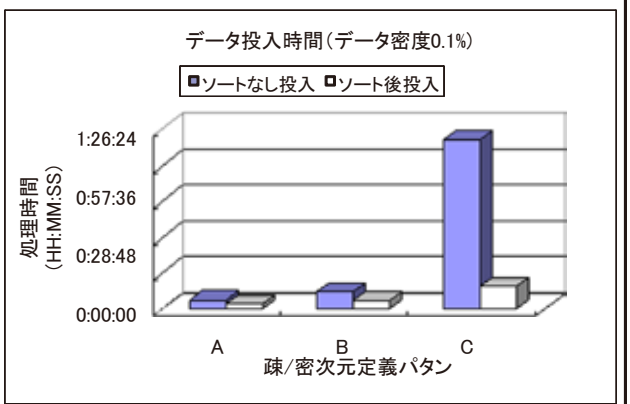
データ投入完了時点でのデータ・ブロック数、非データセル発生状況は、疎/密次元定義パターンに応じ以下の要素を算出していけば事前シミュレーションが可能である。

【データ・ブロックを構成するセル数】・・・①

図４に示す物理構造において、各データ・ブロックを構成するセル数を、図７に示すように「密」定義した次元の項目数の乗算により算出する。

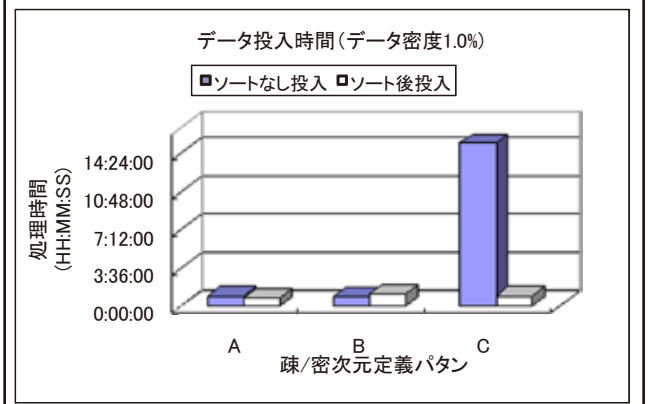
【各データ・ブロック内に入る実データセル数】・・・②

多次元データベースの実データセルは、リレーショナ



疎密パタン	ソートなし投入 (HH:MM:SS)	ソート後投入 (HH:MM:SS)
A	0:03:42	0:02:03
B	0:06:48	0:03:12
C	1:07:59	0:09:06

図５　データ密度0.1% 事前ソートの効果



疎密パタン	ソートなし投入 (HH:MM:SS)	ソート後投入 (HH:MM:SS)
A	0:59:12	0:45:29
B	0:55:24	1:10:47
C	15:20:04	0:58:01

図６　データ密度1.0% 事前ソートの効果

```
データ･ブロックを構成するセル数
　＝（密定義次元1の項目数）x（密定義次元2の項目数）･･･
x（密定義次元nの項目数）
```

図７　データ・ブロックを構成するセル数

ル・データベース実績表の１レコードに対応するという特徴に着目すると、データ投入後、各データ・ブロックに配置されるデータセル数は、リレーショナル・データベース上の実績表に対して図８に示すSQLを発行すれば算出できる。このSQLは実績値を有する疎定義次元項目の組み合わせごとに実績件数を集約して１行の結果を生成する。つまり、結果行の１行ずつがキューブを構成するデータ・ブロックに対応し､各結果行に含まれるCELLSの値は各データ・ブロック内に配置される実データセルの個数に相当する。

```
SELECT
疎定義次元列1,疎定義次元列2,･･･疎定義次元列N, COUNT(*) CELLS
FROM
実績表
GROUP BY
疎定義次元列1,疎定義次元列2,･･･疎定義次元列N
```

図８　各データ・ブロック内に入る実データセル数

【キューブを構成するデータ・ブロック数､キューブ内実データセル総数】・・・③

図８のSQL実行結果として返ってくる結果行から、(1)疎定義次元項目の組み合わせの総和を行ったものが、データ投入時点でのデータ・ブロック数となり、(2)実データセル数CELLSの総和を行ったものが、投入データ件数となる。これらは図９に示すSQLで算出できる。

```
SELECT COUNT(*) BLOCKS,SUM(CELLS) S_CELLS
FROM
(
図8のSQL
)
```

図９　データ・ブロック数と実データセル総数

このSQLの実行結果として返ってくるBLOCKSの値がデータ・ブロック数、S_CELLSの値が、実データセル数(=投入データ件数)に相当する。

【キューブ内非データセル総数】

データ投入時点での非データセル総数は、

（①で算出したデータ・ブロックを構成するセル数）x（③で算出したキューブ内データ・ブロック数）－（投入データ件数）

により算出できる。

引き続き、テストデータを使用してこれら算出方法の妥当性を検証した結果を述べる。

4.2.2. 事前シミュレーション手法の検証

前章の疎/密次元定義パターンによる集計処理テストに使用したベンチマークデータをリレーショナル・データベースに格納、前節の算出方法を適用して、その正当性を確認した。

検証例として、データ密度1.0%の実績データを実績表としてリレーショナル・データベースに格納し、表２ 疎密パターンBに対して、図９のモデルのSQLを適用した結果を図10に示す。

発行SQL

```
SELECT COUNT(*) BLOCKS,SUM(CELLS) S_CELLS
FROM
(
SELECT STORE,CODE,COUNT(*) CELLS
FROM FACT
GROUP BY STORE,CODE
)
```

実行結果

```
BLOCKS    S_CELLS
----------  ----------
660180     12393000
```

図10　疎密パターンB　SQL実行結果　（データ密度1.0% ）

図中FACTは実績表、STORE,CODEは「疎」定義したCustomer次元,Product次元の基本項目が格納される列である。

データ・ブロックを構成するセル数は、「密」定義した次元が Channel、Timeであるから

10(Channel次元定義項目数) x 34(Time次元定義項目数) =340

データセル総数は、

340(データ・ブロックを構成するセル数)　x 660,180

(図10 SQL実行結果のデータ・ブロック数) = 224,461,200

非データセル総数は、

224,461,200(データセル総数) - 12,393,000(投入データ件数) = 212,068,200

この結果は、表４内 データ密度1.0%での疎密パターンBの結果と一致する。

疎密パターンA,Cの場合に対しても、またデータ密度0.1%のデータに対しても同様に非データセル総数の算出を行い、表４の試験結果との完全一致を得た。

これにより、データ投入後のキューブの状態は、4.2.1で述べた算出方法でシミュレーション可能であることを確認することができた。

4.2.3. 事前シミュレーション手法活用指針

実システムにおいては、操作性の観点から多次元キューブの構成次元数は数個に抑えることが実用的である。また、投入データは「疎」な構造となる。投入データが「疎」である場合、「密」次元を増やすと、大量の非データセルを発生させる確率が高くなり、結果的に予期せぬパフォーマンス障害を招きかねない。したがって、どの次元を「疎」とし、「密」とするかの決定に際しては、１つもしくは２つの次元を「密」とし、前節で述べた算出を疎/密次元定義に応じて行い、キューブ構築時発生する非データセルの量を抑制できる組み合わせを選択すればよい。

以上、本章ではデータ投入、事前集計処理についての性能最適化の指針を述べ、テストデータで確認した有効性を示した。引き続き、次章にて性能最適化指針の有効性を実データで確認した事例を示す。

5. 実データでの検証

5.1. 実データ仕様

実データは、３次元の在庫管理モデルに基づくものであり、投入データ件数は481,602件である。データ仕様を表６に示す。

表６　実データ仕様

次元データ			
次元名称	基本項目数	集計項目数	項目数合計
Time	947	41	988
Hinm	17,887	184	18,071
Yard	20	4	24
実績データ			
データ密度(%)		データ件数	
0.14		481,602	

次元軸項目数からデータ密度を算出すると、0.14%となり、投入データは、「疎」な構造となっている。

5.2. 事前シミュレーション

前章での指針に基づき、いずれか１つの次元軸を「密」定義する想定で、考えられる疎/密次元定義のすべての組み合わせに対し、データ投入時点の多次元キューブの状態をリレーショナル・データベース上でシミュレーションした結果を表７に示す。表中①、③の記号は4.1節で述べた算出方法に記載した記号に対応させて記載した。

①データ・ブロック構成セル数は、「密」定義次元の項目数を掛け合わせたものであるから、今回の場合は「密」定義次元の項目数となる。③データ・ブロック総数は、多次

表７　疎/密次元定義シミュレーション結果

疎密パタン	次元名称			データ･ブロック構成セル数	データ･ブロック総数	非データセル総数
	Time	Hinm	Yard	①	③	①x ③ - (投入データ件数)
a	D	S	S	988	21,900	21,155,598
b	S	D	S	18,071	1,905	33,943,653
c	S	S	D	24	236,652	5,198,046

(注): D: 密 (Dense) , S: 疎 (Sparse)

元データベースへの投入データをリレーショナル・データベースに実績表として投入し、前章の図８、図９に従ってSQLを発行して求めた。

表７に示すように、リレーショナル・データベース上でのシミュレーションでは、疎/密次元定義パターンcの場合が、非データセルの発生を最も抑制できるという結果を得た。

5.3. 多次元データベース上での処理結果

表６に示す疎密次元定義の組み合わせすべてに対し、キューブを定義、データ投入を行い、多次元データベース上で集計処理を行った結果を表８に示す。なお、投入データは「疎」とした次元に沿って事前ソートしたものを使用した。同表には、参考として、すべての次元を「密」指定した場合の結果も疎密パターンdとして掲載した。

表８　実データ処理結果

疎密パタン	次元名称			合計処理時間
	Time	Hinm	Yard	(MM:SS)
a	D	S	S	10:42
b	S	D	S	9:08
c	S	S	D	5:23
d	D	D	D	57:41

(注): D: 密 (Dense) , S: 疎 (Sparse)

表中、合計処理時間はデータ投入と集計処理に要した時間の合計である。

最も高速に処理を完了できたのは、疎/密次元定義パターンcの場合であり、これはリレーショナル・データベース上でのシミュレーションで得られた、最適な疎/密次元定義パターンと一致した。

これにより、４章で述べた性能最適化指針の有効性を実データでも確認することができた。

6. おわりに

実システムにおいては、多次元データベース内の実データは多くの場合「疎」な状態となる。そのため、非データセルの発生をいかにうまく阻止できるよう疎/密次元定義を行えるかが性能設計上のポイントとなる。本稿では、疎/密次元定義においてはリレーショナル・データベース上で簡易なSQLでデータ投入後の多次元データベースの状態を事前シミュレーションすることが有効であることを提案し、その手法の有効性を実データでの検証例をもとに紹介した。

多次元データベースの統一的な規格は存在しないため、採用する製品によっては別の観点からの検討も必要となるであろうが、本稿の内容を、MOLAP製品の検討、多次元データベース設計、構築の際の参考にしていただければ幸いである。

参考文献

1) Codd E.F., Codd S.B., and Salley C.T.," Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate",Codd & Date, Inc(1993)
2) 谷村啓匠ほか編集,"データウェアハウス活用ツール選定ガイド",㈱アイ・ティ・アール(1999)
3) 一色淳夫ほか," MOLAPのための多次元配列実現方式", 電子情報通信学会 第13回データ工学ワークショップ論文集(2002)
