Data Structures for Range-Sum Queries

The Evolution of the Data Cube

Paul Butler

CUMC 2010
Waterloo, Ontario

July 2010

The Range-Sum Problem: Example

Name	DOB	City	Height	Siblings	Pets
Joseph Matthews	Jan 9, 1987	Waterloo	172	2	1
Sarah MacDonald	Nov 17, 1988	Halifax	167	0	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- How many people in Vancouver were born before 1990?

The Range-Sum Problem: Example

Name	DOB	City	Height	Siblings	Pets
Joseph Matthews	Jan 9, 1987	Waterloo	172	2	1
Sarah MacDonald	Nov 17, 1988	Halifax	167	0	2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- How many people in Vancouver were born before 1990?
- What is the average number of siblings for people above 170 cm?

The Range-Sum Problem: Example

Name	DOB	City	Height	Siblings	Pets
Joseph Matthews	Jan 9, 1987	Waterloo	172	2	1
Sarah MacDonald	Nov 17, 1988	Halifax	167	0	2
⋮	⋮	⋮	⋮	⋮	⋮

- How many people in Vancouver were born before 1990?
- What is the average number of siblings for people above 170 cm?
- What is the total number of pets owned by people 18 to 24 in Calgary?

The Range-Sum Problem: Definitions

Definition (Measure)

A column whose values we want to aggregate in our queries.

The Range-Sum Problem: Definitions

Definition (Measure)

A column whose values we want to aggregate in our queries.

Example

The **Pets** and **Siblings** columns from the last example are **measures**

The Range-Sum Problem: Definitions

Definition (Measure)

A column whose values we want to aggregate in our queries.

Example

The **Pets** and **Siblings** columns from the last example are **measures**

Definition (Dimension)

A column we want to use to select columns which belong to our aggregation.

The Range-Sum Problem: Definitions

Definition (Measure)

A column whose values we want to aggregate in our queries.

Example

The **Pets** and **Siblings** columns from the last example are **measures**

Definition (Dimension)

A column we want to use to select columns which belong to our aggregation.

Example

The **DOB**, **City**, and **Height** columns are **dimensions**.

Dimensions vs. Measures

- For each column, **dimension** vs. **measure** depends on which questions you want to answer.

Dimensions vs. Measures

- For each column, **dimension** vs. **measure** depends on which questions you want to answer.
- *What is the average age of people with two pets*

Dimensions vs. Measures

- For each column, **dimension** vs. **measure** depends on which questions you want to answer.
- *What is the average age of people with two pets*

Dimensions vs. Measures

- For each column, **dimension** vs. **measure** depends on which questions you want to answer.
- *What is the average age of people with two pets*
  - **DOB** would be a **measure**
  - **Pets** would be a **dimension**

The Naïve Approach

- Store the data as a table
- For every row:
  - If the dimension columns match our query, add the value in the measure column to a running total
- Return the running total

The Naïve Approach

- Store the data as a table
- For every row:
  - If the dimension columns match our query, add the value in the measure column to a running total
- Return the running total

Too slow for large amounts of data!

Data Cubes

We can aggregate the data into a multi-dimensional array. [3]

Pets		DOB						
		...	1985	1986	1987	1988	1989	...
Height	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱
	165	...	192	342	558	56	591	...
	166	...	325	275	707	855	484	...
	167	...	487	326	363	193	350	...
	168	...	326	363	193	350	422	...
	169	...	438	456	550	385	412	...
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱

Data Cubes

We can aggregate the data into a multi-dimensional array. [3]

Pets		DOB						
		...	1985	1986	1987	1988	1989	...
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱
	165	...	192	342	558	56	591	...
	166	...	325	275	707	855	484	...
Height	167	...	487	326	363	193	350	...
	168	...	326	363	193	350	422	...
	169	...	438	456	550	385	412	...
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱

Now cells not selected by the query are ignored.
Less lookups, faster queries (but still too slow).

Data Cubes

We can calculate partial sums for each row, column, etc. [3]

Pets		DOB							
		...	1985	1986	1987	1988	1989	...	Sum
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	165	...	192	342	558	56	591	...	10937
	166	...	325	275	707	855	484	...	10998
Height	167	...	487	326	363	193	350	...	11064
	168	...	326	363	193	350	422	...	10913
	169	...	438	456	550	385	412	...	11347
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	Sum	...	8121	8255	8206	8820	8026	...	202169

Data Cubes

We can calculate partial sums for each row, column, etc. [3]

Pets		DOB							
		...	1985	1986	1987	1988	1989	...	Sum
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	165	...	192	342	558	56	591	...	10937
	166	...	325	275	707	855	484	...	10998
Height	167	...	487	326	363	193	350	...	11064
	168	...	326	363	193	350	422	...	10913
	169	...	438	456	550	385	412	...	11347
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	Sum	...	8121	8255	8206	8820	8026	...	202169

Queries which only mention a subset of the dimensions run faster.

Data Cubes

Example

How many pets are owned by people born between 1986 and 1988 (inclusive)?

Pets		DOB							
		...	1985	1986	1987	1988	1989	...	Sum
Height	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	165	...	192	342	558	56	591	...	10937
	166	...	325	275	707	855	484	...	10998
	167	...	487	326	363	193	350	...	11064
	168	...	326	363	193	350	422	...	10913
	169	...	438	456	550	385	412	...	11347
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	Sum	...	8121	**8255**	**8206**	**8820**	8026	...	202169

Data Cubes

Example

How many pets are owned by people born between 1986 and 1988 (inclusive)?

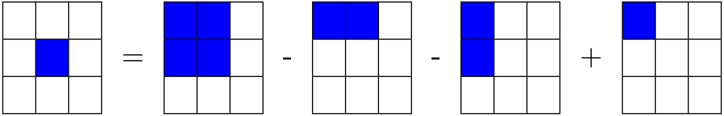
Pets		DOB							
		...	1985	1986	1987	1988	1989	...	Sum
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	165	...	192	342	558	56	591	...	10937
	166	...	325	275	707	855	484	...	10998
Height	167	...	487	326	363	193	350	...	11064
	168	...	326	363	193	350	422	...	10913
	169	...	438	456	550	385	412	...	11347
	⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋱	⋮
	Sum	...	8121	**8255**	**8206**	**8820**	8026	...	202169

$8255 + 8206 + 8820 = \mathbf{25281}$

With **Data Cubes**, the question becomes
How do we find the sum of values which fall inside a given (hyper)rectangle in a multidimensional array?

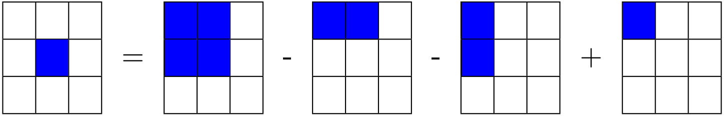
Prefix-Sum Table

Observation: range sums can be computed as a sum of four range queries starting from 0. [4]



Prefix-Sum Table

Observation: range sums can be computed as a sum of four range queries starting from 0. [4]



In general: $\leq 2^d$ operations, where d is number of dimensions.

With **Prefix-Sum Tables**, the question becomes
How do we find the sum of values which fall inside a given (hyper)rectangle in a multidimensional array with one corner fixed at the origin?

Prefix-Sum Table

Array A

Index	0	1	2	3	4	5	6	7	8
0	3	5	1	2	2	4	6	3	3
1	7	3	2	6	8	7	1	2	4
2	2	4	2	3	3	3	4	5	7
3	3	2	1	5	3	5	2	8	2
4	4	2	1	3	3	4	7	1	3
5	2	3	3	6	1	8	5	1	1
6	4	5	2	7	1	9	3	3	4
7	2	4	2	2	3	1	9	1	3
8	5	4	3	1	3	2	1	9	6

Array P

Index	0	1	2	3	4	5	6	7	8
0	3	8	9	11	13	17	23	26	29
1	10	18	21	29	39	50	57	62	69
2	12	24	29	40	53	67	78	88	102
3	15	29	35	51	67	86	99	117	133
4	19	35	42	61	80	103	123	142	161
5	21	40	50	75	95	126	151	171	191
6	25	49	61	93	114	154	182	205	229
7	27	55	69	103	127	168	205	229	256
8	32	64	81	116	143	186	224	257	290

[1]

Updating the Prefix-Sum Table

Array A

Index	0	1	2	3	4	5	6	7	8
0	3	5	1	2	2	4	6	3	3
1	7	* 4	2	6	8	7	1	2	4
2	2	4	2	3	3	3	4	5	7
3	3	2	1	5	3	5	2	8	2
4	4	2	1	3	3	4	7	1	3
5	2	3	3	6	1	8	5	1	1
6	4	5	2	7	1	9	3	3	4
7	2	4	2	2	3	1	9	1	3
8	5	4	3	1	3	2	1	9	6

Array P

Index	0	1	2	3	4	5	6	7	8
0	3	8	9	11	13	17	23	26	29
1	10	* 19	22	30	40	51	58	63	70
2	12	25	30	41	54	68	79	89	103
3	15	30	36	52	68	87	100	118	134
4	19	36	43	62	81	104	124	143	162
5	21	41	51	76	96	127	152	172	192
6	25	50	62	94	115	155	183	206	230
7	27	56	70	104	128	169	206	230	257
8	32	65	82	117	144	187	225	258	291

[1]

Relative Prefix Method

Geffner, Agrawal, El Abbadi, Smith [1] introduced two new tables

- relative-prefix table
- overlay table

Overlay Table - Anchor Value

Index	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6				V					
7									
8									

[1]

Overlay Table - Outer Values

Index	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6					X_1				
7				Y_1					
8									

Index	0	1	2	3	4	5	6	7	8
0									
1									
2									
3									
4									
5									
6						X_2			
7									
8				Y_2					

[1]

Relative Prefix Table

Relative Prefix (RP) array

Index	0	1	2	3	4	5	6	7	8
0	3	8	9	2	4	8	6	9	12
1	10	18	21	8	18	29	7	12	19
2	12	24	29	11	24	38	11	21	35
3	3	5	6	5	8	13	2	10	12
4	7	11	13	8	14	23	9	18	23
5	9	16	21	14	21	38	14	24	30
6	4	9	11	7	8	17	3	6	10
7	6	15	19	9	13	23	12	16	23
8	11	24	31	10	17	29	13	26	39

[1]

Relative Prefix Method

Array A

Index	0	1	2	3	4	5	6	7	8
0	3	5	1	2	2	4	6	3	3
1	7	3	2	6	8	7	1	2	4
2	2	4	2	3	3	3	4	5	7
3	3	2	1	5	3	5	2	8	2
4	4	2	1	3	3	4	7	1	3
5	2	3	3	6	1	8	5	1	1
6	4	5	2	7	1	9	3	3	4
7	2	4	2	2	3	1	9	1	3
8	5	4	3	1	3	2	1	9	6

Overlay boxes of size 3x3

Index	0	1	2	3	4	5	6	7	8
0	0	0	0	9	0	0	17	0	0
1	0			12			33		
2	0			20			50		
3	12	12	17	46	13	27	97	10	24
4	0			7			17		
5	0			15			40		
6	21	19	29	86	20	51	179	20	40
7	0			8		*	14		
8	0			20			32		

Relative Prefix (RP) array

Index	0	1	2	3	4	5	6	7	8
0	3	8	9	2	4	8	6	9	12
1	10	18	21	8	18	29	7	12	19
2	12	24	29	11	24	38	11	21	35
3	3	5	6	5	8	13	2	10	12
4	7	11	13	8	14	23	9	18	23
5	9	16	21	14	21	38	14	24	30
6	4	9	11	7	8	17	3	6	10
7	6	15	19	9	13	* 23	12	16	23
8	11	24	31	10	17	29	13	26	39

[1]

Updating Overlay and Relative Prefix Table

Index	0	1	2	3	4	5	6	7	8
0									
1		*							
2									
3									
4									
5									
6									
7									
8									

Figure 14. Overlay regions during an update.

Relative Prefix (RP) array

Index	0	1	2	3	4	5	6	7	8
0	3	8	9	2	4	8	6	9	12
1	10	* 19	22	8	18	29	7	12	19
2	12	25	30	11	24	38	11	21	35
3	3	5	6	5	8	13	2	10	12
4	7	11	13	8	14	23	9	18	23
5	9	16	21	14	21	38	14	24	30
6	4	9	11	7	8	17	3	6	10
7	6	15	19	9	13	23	12	16	23
8	11	24	31	10	17	29	13	26	39

Overlay boxes

Index	0	1	2	3	4	5	6	7	8
0	0	0	0	9	0	0	17	0	0
1	0	*		13			34		
2	0			21			51		
3	12	13	18	47	13	27	98	10	24
4	0			7			17		
5	0			15			40		
6	21	20	30	87	20	51	180	20	40
7	0			8			14		
8	0			20			32		

[1]

Dynamic Data Cube

Root:

Q			11	R			15
			29				33
			40				48
15	29	35	**51**	16	35	**48**	66
S			10	T			15
			24			*	31
			42				47
12	26	34	52	8	30	54	61

T:

U	7	V	8
4	**16**	**12***	15
W	10	Z	6
4	14	12	16

V:

7	1
5*	2

[2]

Δ-tree

Steve Geffner, Divyakanth Agrawal, Amr El Abbadi, and Terry Smith.
Relative prefix sums: An efficient approach for querying dynamic olap data cubes.
Technical report, Santa Barbara, CA, USA, 1999.

Steven Geffner, Divyakant Agrawal, and Amr El Abbadi.
The dynamic data cube.
In Carlo Zaniolo, Peter C. Lockemann, Marc H. Scholl, and Torsten Grust, editors, *EDBT*, volume 1777 of *Lecture Notes in Computer Science*, pages 237–253. Springer, 2000.

Jim Gray, Adam Bosworth, Andrew Layman, Don Reichart, and Hamid Pirahesh.
Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals.
pages 152–159, 1996.

Ching-Tien Ho, Rakesh Agrawal, Nimrod Megiddo, and Ramakrishnan Srikant.
Range queries in olap data cubes.
SIGMOD Rec., 26(2):73–88, 1997.

Slides

`github.com/paulgb/cumc2010/raw/master/slides.pdf`

Try Sage!

`sagemath.org`