More Topics in OLAP

Topics:

- Multidimensional operations
- Iceberg cubes
- Indexing for ROLAP
- Computing the Datacube

Multidimensional Operations

See [PJ01, PR99, B$^+$04]

Beware: Everyone uses the same terms, but many different definitions (esp. for “slice” and “dice”).

Overview

Operators: drill-down, roll-up, slice-and-dice, pivot

Issues for each:

▷ what it does

▷ implementing using SQL and star schema

First: another look at SQL aggregation

Aggregation in SQL

Table junk:

store	item	sales
Moncton	flour	300
Fton	flour	150
Moncton	milk	10
Fton	sugar	100

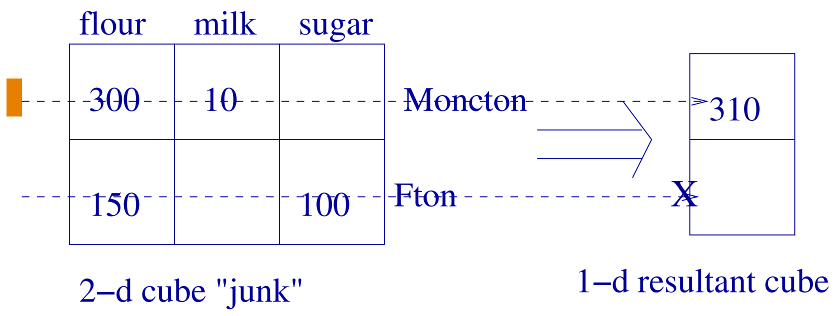
SQL Query:

```
select store,sum(sales)
from junk
group by store
having count(*) > 1
and sum(sales) > 300
```

Resulting table:

store	sum(sales)
Moncton	310

Multidimensional viewpoint



Relational Example: Registrar

Consider the join ($\bowtie$) of Fact table and Dim tables. Corresponds to a 2-d cube.

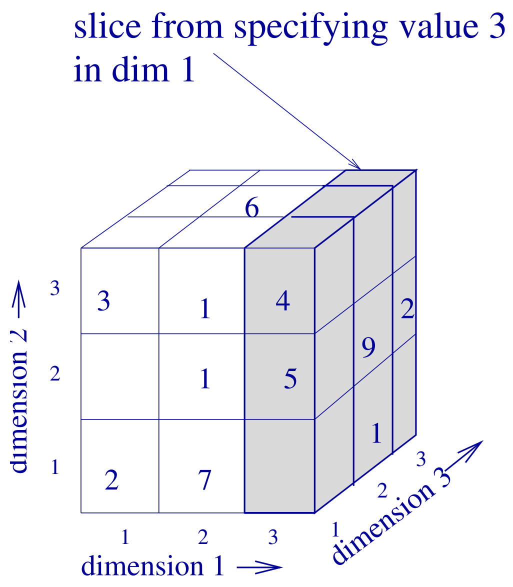
Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
CS5678	CSCourses	Science	Al	u/g	73
EE2222	EECourses	Engg	Jack	grad	25
EE2222	EECourses	Engg	Al	u/g	95
CS4002	CSCourses	Science	Jack	grad	89
CS4002	CSCourses	Science	Al	u/g	62

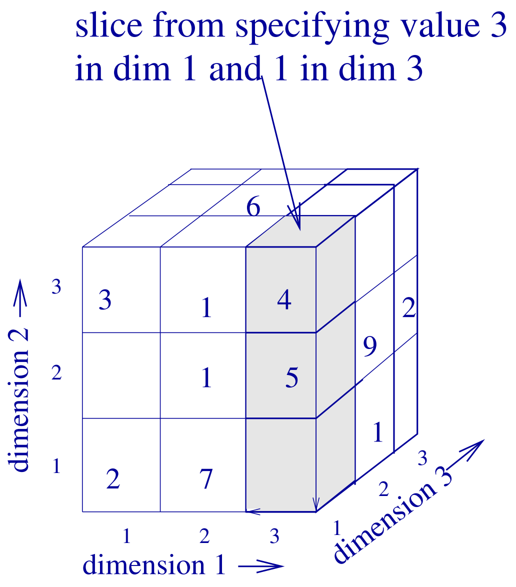
Same number of rows as fact table

Slicing Operation

Slicing takes a d-dim cube, returns a $(d-k)$-dim cube

We specify one fixed value for each of k dimensions. (In [PR99] your "fixed values" must be $\top$.)





Slicing in SQL

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
. . .					

Take Jack’s slice (finest granularity for courses):

```
select course, avg(grade)
from junk
where student = 'Jack'
group by course,discipline,faculty
```

Result has measure and info on Course dimension. Like 1-d cube.

Slicing in SQL(2)

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
. . .					

Take grads' slice (lowest granularity for courses):

```
select course, avg(grade)
from junk
where kind = 'grad'
group by course,discipline,faculty
```

(This makes sense if you view grad as a particular Student value. If you view it as a *collection* of individual students, it's not a slice. . .)

Slicing in SQL (3)

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
...					

Take Jack's slice and (intermediate granularity for Course):

```
select discipline,avg(grade)
from junk
where student = 'Jack'
group by discipline,faculty
```

Result still has measure and info on Course dimension. Higher granularity. Note we **dropped** a group-by. We did a rollup on the course dimension.

Rollup and Drill Down

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
. . .					

Rollup: shifting to coarser granularity (higher in dim. hierarchy).

Drill-down: reverse shift. “Gimme more detail”

Rollup Example

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
...					

Changing query from

```
select
course,student,avg(grade)
from junk
group by course, discipline,
faculty, student, kind
```

to

```
select
discipline,kind,avg(grade)
from junk
group by
discipline,faculty,kind
```

Rollup Example (2)

Going left to right is a rollup on both Course and Student and going from the right query to the left is a drill-down on both dims

“Extreme” Rollup

Recall $\top$ represents **all values** of a dimension.

Rollup to this level means “**don’t** group”.

“Extreme” Rollup(2)

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
. . .					

eg, from

```
select
discipline,kind,avg(grade)
from junk
group by
discipline,faculty,kind
```

to

```
select kind, avg(grade)
from junk
group by kind
```

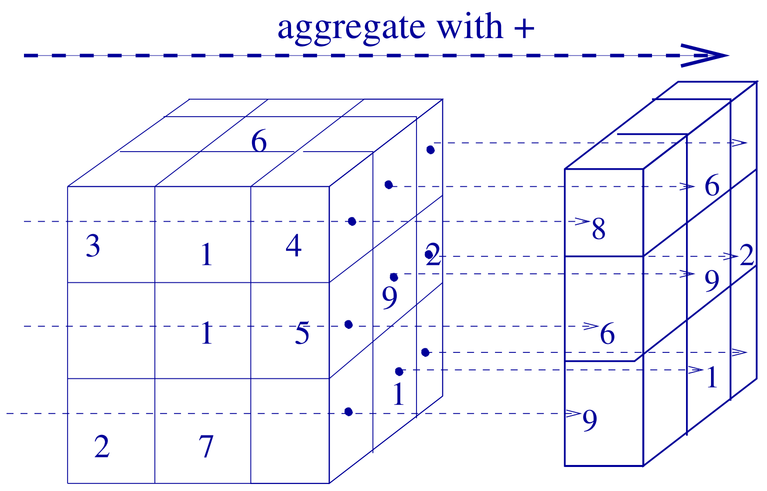
We’re “partly rolled up” on Students. Total rollup on Course.

Cube view of “extreme rollup”

Given d-dimensional cube

Complete rollup on k dims: get $(d-k)$-dimensional result.

Like slicing, except we definitely aggregate.

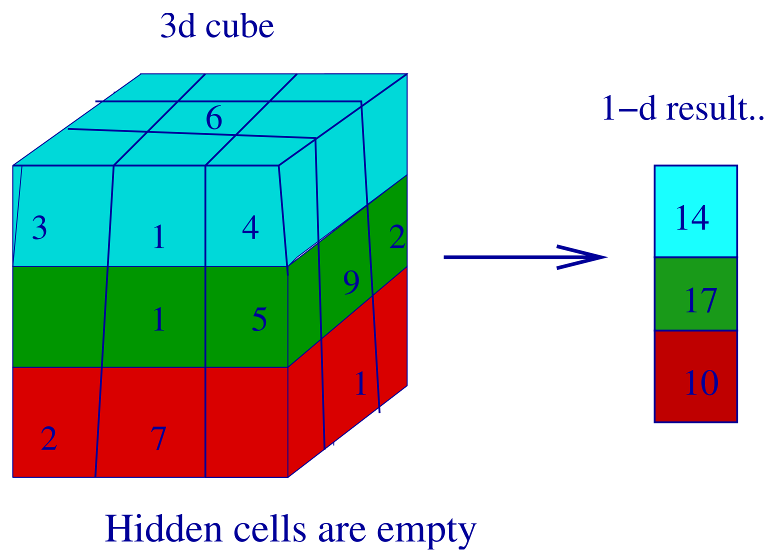


Hidden cells are empty

(This is what [PR99] calls “slicing” and what [B$^+$04] calls “dicing” !!)

Cube view of “extreme rollup” (2)

Can aggregate along several dimensions.



Rollup in SQL

SQL now has a ROLLUP feature used with GROUP BY. Computes a single result table for a collection of related GROUP-BYs. (Every prefix of a list).

Use: list should contain the different levels of one dimension, by decreasing granularity.

Eg, list faculty, discipline, course for Course dim.

ROLLUP will lead to 4 group-bys: (*nothing*), (faculty), (faculty,discipline), (faculty, discipline, course)

Syntax: What’s the ANSI standard for ROLLUP?

Dunno if the ANSI SQL standard changed, or whether different vendors just extended SQL their own way, but . . .

With MySQL and (I think) SQLServer, just add modifier WITH ROLLUP after the group-by list. eg

```
SELECT *,avg(grade) from junk
GROUP BY faculty, discipline, course WITH ROLLUP
```

With Oracle and others, last line becomes

```
GROUP BY ROLLUP(faculty,discipline, course)
```

Output has NULLs

```
SELECT faculty, discipline, course, avg(grade) from junk
GROUP BY faculty, discipline, course WITH ROLLUP
```

gives output with NULLs

(if you GROUP BY `faculty`, how can you expect to have `course`?)

Output has NULLs (2)

	Course dim		Measure
course	**discipline**	**faculty**	**avg(grade)**
CS5678	CSCourses	Science	72.3
. . .			
CS4444	CSCourses	Science	95.3
NULL	CSCourses	Science	67.2
. . .			
NULL	PhysCourses	Science	69.2
NULL	NULL	Science	77.2
. . .			
NULL	NULL	Arts	87.2
NULL	NULL	NULL	81.2

Dicing: My idea

People usually talk about “slice-and-dice” together. They start to disagree when asked to describe “dice” or “slice” alone!

My view follows

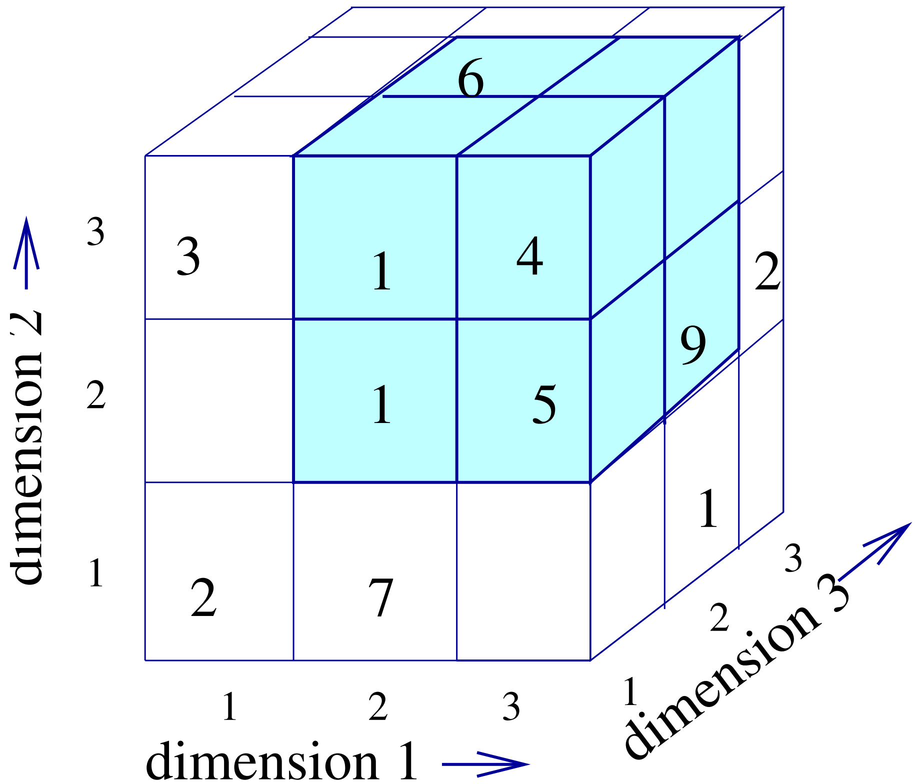
Dicing refers to **range selection** in multiple dimensions.

Eg, select range 2-3 for dims 1 and 2,

select range 1-2 for dim 3.

Dicing as range selection

Note that the dimensionality is not reduced.



Dicing: Idea in [PR99]

Dicing is specifying certain values to exclude in one dimension.

Need not form a range.

(Specifying which ones to keep is their select operation.)

Previous example,

Dice(dim1, {1}, (Dice (dim2, {1}, (Dice (dim3,{3}, theCube)))))

Actually, some predicate can *test* the dimension values

Dicing in SQL, example

Dicing in SQL would not involve aggregation[1] but would involve WHERE.

Course dim			Student dim		Measure
course	**discipline**	**faculty**	**student**	**kind**	**grade**
CS5678	CSCourses	Science	Jill	grad	87
...					

```
select * from junk where course between 'CS3300' and 'CS6789'
and student between 'James' and 'Karen'
```

[1]Assume that it is always done at the finest granularity for a dimension.

Pivot and CrossTabs

Some operations are concerned with information display.

Pivot: choose which dimensions to show in a (usually) 2-d rendering.

Cross-tabs: spreadsheet-style row/column aggregates.

Pivot

Pivot deals with presentation $\Rightarrow$ no SQL counterpart.

Choose some dimensions $X_1, \ldots, X_i$ to appear on x and some dims $Y_1, \ldots, Y_j$ to appear on y.

Aggregate as in $\ldots$ GROUP BY $X_1, \ldots, X_i, Y_1, \ldots Y_j$. Resulting values stored in the x, y grid.

Pivot, example 1

Registrar cube:

$$\mathrm{Session} \times \mathrm{Student} \times \mathrm{Course} \times \mathrm{Section} \times \mathrm{Campus} \times \mathrm{ForCredit} \longrightarrow \mathrm{Grade}$$

Pivot choosing Student for x and Session for y

	Jill	Jack	Al
Spring’03		42	76
Summer’03	87	89	20

Pivot, showing > 2 dimensions

You can pack several dimensions onto one display axis.

(Typically, by Cartesian product).

Pivot choosing Student and ForCredit for x and Session for y

	Jill		Jack		Al	
	Yes	No	Yes	No	Yes	No
Spring'03		42			76	
Summer'03	86	88		89	38	2

Pivot with CrossTabs example

	Jill	Jack	Al	Avg
Spring'03		42	76	**47.8**
Summer'03	87	89	20	39.4
Avg	87	51.4	29.3	42.9

Other OLAP operations: Push and Pull

Pourabbas and Rafanelli [PR99] discuss push and pull operations to *interchange dimensions and measures.*

Thinking about such interchange is quite natural in the ROLAP setting. (Why are some columns treated differently than others?)

To be useful as dimensions, you'd probably want to "bucket" real-valued measures into discrete ranges.

Eg, `grade` converted to A, B-, C, etc.

Other OLAP operations: Drill Across

According to [B+04]. . . Recall:

Drill down: go to finer granularity. “Towards your kids” in the hierarchy.

Roll up: go to coarser granularity. “Towards your parent”.

So, maybe drill across should let you navigate amongst your siblings, cousins, etc. ▮? Traverse the slices for the various values at the same hierarchical level?

Eg: from CSCourses to EECourses to PhysCourses to . . .

▮

Another drill-across

Pedersen and Jensen[PJ01] suggest that drill across instead is like doing a join between two cubes sharing some common dimensions.

“Across” means from one cube, to another.

Eg, from Jill’s slice in the Registrar cube, to Jill’s slice in the ParkingTicket cube.

Combining measures

We may need ways to create new measures. Eg, got a `Cost` measure and a `Benefit` measure; want to compute $\frac{\text{Cost}}{\text{Benefit}}$

Computing the datacube

Recall, the datacube is “all possible GROUP BYs”.

SQL now has a standard way to compute the datacube. CUBE keyword use is similar to ROLLUP.

SELECT *, avg(m) FROM junk GROUP BY CUBE(a,b,c,d)

or (other dialects)

SELECT *, avg(m) FROM junk GROUP BY a,b,c,d WITH CUBE

ROLLUP would have $4+1$ GROUP BYs; CUBE would have 2^4

NULLs or ALLs in the table

As with SQL's ROLLUP, we get NULLs in the resulting table.

Or, some databases may use ALLs in place of $\top$.

ALL seems more logical. But having multiple "special values" may be painful to DB implementors. You've already forced to have a special NULL value, so reuse it!

Other operations

Many OLAP cubes have at least one time dimension.

Time-series analysis often provided.

- ▷ Trailing averages,
- ▷ transform to frequency domain, etc.

percentiles and histogram generation are useful too.

Note sure about ANSI SQL, but many vendor extensions do such things.

Iceberg Cubes

Consider view V with nearly as many measures as source cube S.

Most measures of V are useless for data mining, where “support” is important.

So, make smaller “Iceberg cube”[BR99]:

IF measure value in V is *not* aggregate of $\geq M$ measures from S

THEN discard it.

SQL: like adding `HAVING COUNT(*) >= M`

Guarantees V has $\leq 1/M$ measures as S.

In V, most measures of S were *not* aggregated with any other measures.

A Few ROLAP Issues

- ▷ Indexing for star schema
- ▷ Computing the datacube.

(In OLAP, queries outnumber updates. Materialized views and fancy indexing usually good.)

Indexing for the Star Schema

In relational databases, much effort spent optimizing $\bowtie$.

Fact table joins dimension table via foreign keys only.

Very special case: exploit it! A *join index* links each dimension value to all matching facts.

eg. Index entry for “Jill” (of students table) is a linked list giving all Jill’s fact-table entries.

Multikey Join Index

A *Multikey join index* can precompute an *n*-way join. eg. Index entry for “(Jill, Spring’03)” is a linked list giving all Jill’s fact-table entries for Spring’03 session.

Briefly mentioned in Chaudhuri et al.[C^+01, p. 50].

Bitmap Indexing and Star Schema

Suppose there are few values (say f_i) in dimension i.

Likewise, small f_j in dimension j. Make $f_i + f_j$ **bit vectors** B_k, each as long as fact table.

For $1 \leq k \leq f_i$, $B_k[n] = 1$ iff n^{th} fact has value k in dimension i.

ANDing bitvectors is fast. [B_k stored contiguously]

Issue: consider only the finest granularity for dimension i? I'd say not. Include the "fake" values at higher aggregation levels.

Eg, besides an entry for "Jill", have an entry "grad".

Bitmap Indexing, example

Fact Table			Bitmap Indices		
cID	sID	Grade	B_1	$\ldots$	B_{big}
0	0	87	1		0
0	2	73	0		0
1	1	25	1		1
1	2	95	0		1
2	1	89	1		0
2	2	62	0		0

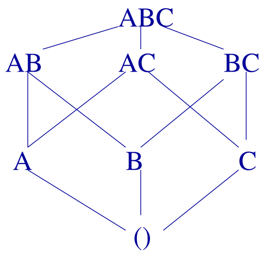
In example, suppose B_1 means “This fact is about grad students”, and B_{big} means “This fact is about EE2222”.

Students 0 and 1 are grad students. EE2222 is course 1. ANDing the two columns finds records about grads taking EE2222.

Computing the Datacube (Efficiently)

Do not compute each view from the source cube. Views can be computed from other views (much more cheaply). Recall the "can be computed from" lattice.

Key: A=Session, B=Course,C=Student



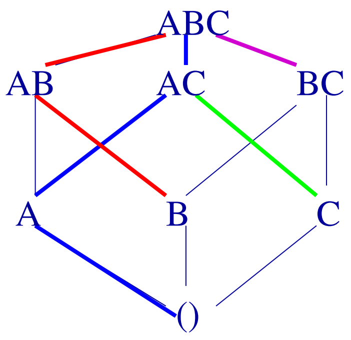
For dense source cube, whole datacube may not be much bigger than source. Sparse cubes are a problem.

ROLAP Algorithms for Datacube

Good reference is Sarawagi et al.[SAG96], presents `PipeSort` and `PipeHash` algorithms.

Focus on PipeSort. It assumes you've decomposed "can-be-computed-from" lattice into a set of paths covering all nodes.

Key: A=Session, B=Course,C=Student

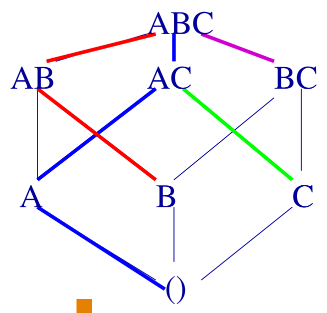


PipeHash is known to be hard on memory.

PipeSort

“Pipeline” along each path. Eg, blue path ABC-AC-A-():

Key: A=Session, B=Course,C=Student



1. sort by ACB; now tuples grouped by AC values.
2. Aggregate, write a tuple to disk AND to a buffer aggregating the current value of A.
3. When the current value of A changes, write tuple to disk AND to a buffer aggregating for ().

Then go for green path AC-C: Resort AC by C,. . .

PipeSort can share sorts

Also can consider *sharing* a sort.

If you already have data sorted by ABCD and want data sorted by ACD, you have less work to do: data already ordered by A, so you just need to do a small (probably in-memory) re-sort for each distinct value of A.

Much faster than doing an external memory (disk) sort on whole data.

PipeSort Example

A	B	C	meas
a_2	b_1	c_1	5
a_2	b_2	c_1	6
a_1	b_1	c_2	5
a_1	b_3	c_2	10
a_1	b_1	c_1	5
a_2	b_2	c_2	15

$\Longrightarrow$

A	C	B	meas
a_1	c_1	b_1	5
a_1	c_2	b_1	5
a_1	c_2	b_3	10
a_2	c_1	b_1	5
a_2	c_1	b_2	6
a_2	c_2	b_2	15

Cost of PipeSort

d-dimensional relation: Pipesort does $\binom{d}{\lceil d/2 \rceil}$ sorts.

Many sorts will be expensive external-memory sorts.

Continuing Work

Late 1990s saw much work in datacube computation:

- ▷ in parallel
- ▷ for iceberg cubes
- ▷ for MOLAP

References

[B+04] C. Boyens et al. *Electronic Handbook of Computational Statistics*, chapter 9. Springer (online at http://www.quantlet.com/mdstat/scripts/csa/pdf/csa.pdf), 2004.

[BR99] Kevin Beyer and Raghu Ramakrishan. Bottom-up computation of sparse and iceberg cubes. In *SIGMOD'99*, pages 359–370, 1999.

[C+01] Surajit Chaudhuri et al. Database technology for decision support systems. *IEEE Computer*, pages 48–55, December 2001.

[PJ01] Torben Bach Pedersen and Christian S. Jensen. Multidimensional database technology. *IEEE Computer*, pages 40–46, December 2001.

[PR99] E. Pourabbas and M. Rafanelli. Characterization of hierarchies and some operations in OLAP environment. In *Proceedings, DOLAP'99*, 1999.

[SAG96] S. Sarawagi, R. Agrawal, and A. Gupta. On Computing the Data Cube, 1996.