

Storage Techniques and ROLAP Schemas

Topics:

- + MOLAP, ROLAP, and HOLAP
- + Snowflake schema (vs. star schema)
- + Level indicator

MOLAP, ROLAP, and HOLAP

One viewpoint: {M,R,H}OLAP are just implementation approaches to provide multidimensional databases.■

- ▷ MOLAP: we know how to store big multi-dim arrays! Do this on disk.■
- ▷ ROLAP: we're good at storing relational data, so store the cube as a relation.■
- ▷ HOLAP: a little of both... more soon.■

MOLAP arrays (6-d university example)

Session \times Student \times Course \times Section \times Campus \times ForCredit \rightarrow Grade



Over last 25 years:

100 Session values, 100,000 Students, 1000 Courses, (up to) 10 Sections, 2 Campuses, 2 ForCredit values.



Big array:

$10^2 \times 10^5 \times 10^3 \times 10 \times 2 \times 2 = 4 \times 10^{11}$ cells. (Maybe 800GB storage)



Sparseness

So, array has 4×10^{11} cells.

Yet each student would take at most 40 courses. Hence, at most 4×10^6 facts.

Density of cube = $\frac{4 \times 10^6}{4 \times 10^{11}} = 0.001\%$.

Our cube is sparse.

MOLAP and Its Advantages

It's *really* quick to retrieve an item from an array, if you know its index. direct mapping■

If the array is dense (not sparse), computer memory is nicely filled with measure values **only** (and a few blanks). The dimensional values for a fact are *implicit* in the address.■

We know array indices are integers.

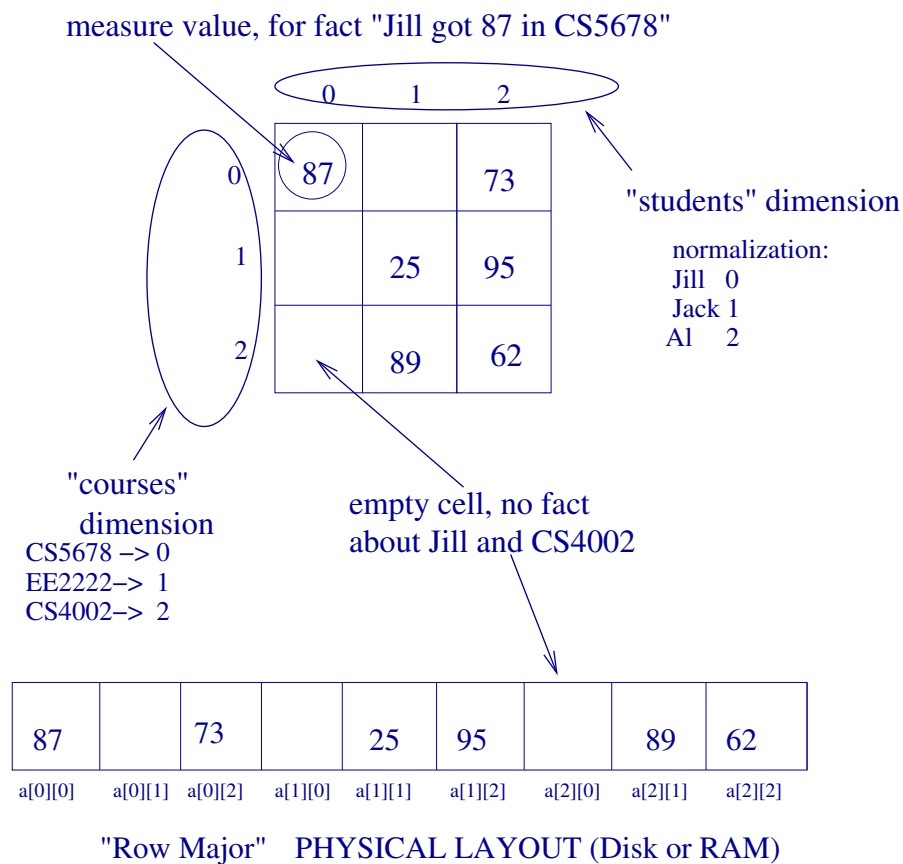
But dimension values are typically strings, “Jack”, “Jill” etc.

Thus dimension values must be *normalized* (mapped to integers).

eg. “Jack”→0, “Jill”→1, etc.



MOLAP Example



MOLAP and Sparseness

Scientific computing has dealt with huge sparse 2-d and 3-d arrays for years.■

Tricks

- ▷ store data in blocks (no more “row-major” arrays)■
- ▷ use data compression carefully
 - some compression techniques look suspiciously like storing relations.

Curse of Dimensionality

Observation: The “Curse of Dimensionality”: techniques that work well for 2-d or 3-d fail miserably on 20-d! ■

Sizes and other costs grow exponentially with the number of dimensions.

ROLAP and Its Advantages

RelationalOLAP stores facts as tuples in relations.■

We've already seen the star schema.■

- ▷ Relational technology's very mature.
- ▷ Good for sparse cubes: no fact, no storage!
- ▷ Need fancy indexing for speed

ROLAP Operations

Operations could be supplied in two ways

- ▷ Augment the standard set of relational operators with some OLAP ones, notably “CUBE-BY” and “ROLLUP” in SQL.■
- ▷ A “ROLAP Middleware” will accept OLAP queries and reformulate them for a relational engine.

HOLAP

ROLAP best on sparse data; MOLAP best on dense data

HOLAP: do both (A **h**ybrid. Maybe some materialized view are dense, but the base cube is sparse.■store the view with MOLAP; store the base cube with ROLAP.■


Carry idea further: Observe that *parts* of a cube/view are dense, even if most of it is sparse.■Store/process the dense parts like MOLAP; process the sparse parts like ROLAP.

HOLAP example

eg 6-d registrar's cube:

most of the facts about specific CS student “James” who started in 1995 will relate to CS courses during the sessions from 1995–1998.■

eg There are **no** facts about the Intro-to-Internet course from sessions before 1995.



Schemas for ROLAP

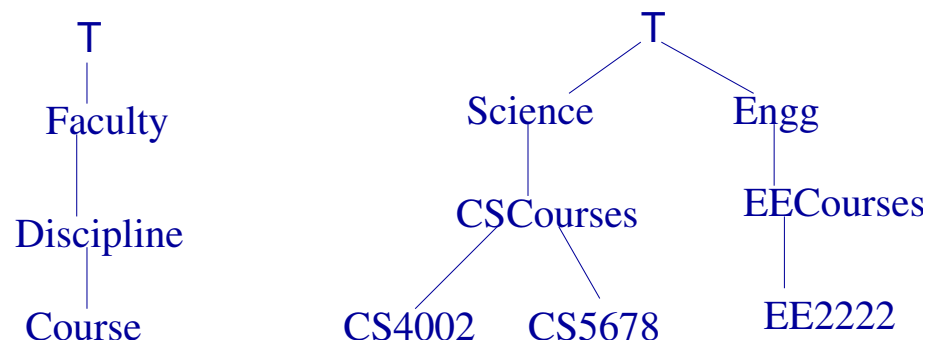
Already we've seen the **star schema** where a central **fact** table has foreign keys into the dimension tables.■

Another variation, with less redundancy, is the “snowflake schema”.

Redundancy in the Star Schema

Let's make the "Courses" dimension more interesting.

Now all courses are contained in some Discipline, and all Disciplines are contained in a Faculty.



Redundancy in the Star Schema, cont.

Using the star schema, the “Courses” table is

cID	Course	Discipline	Faculty
0	CS5678	CSCourses	Science
1	EE2222	EECourses	Engg
2	CS4002	CSCourses	Science

Note the redundancy: every tuple that contains “CSCourses” for Discipline will necessarily contain “Science” for Faculty.

Star-Schema Redundancy, 2.

This sort of redundancy is usually a sign of **bad** relational design.

But DW is special: usual rules may not apply.■

Space waste is not significant: the Fact table dwarfs dimension tables.■

“Update anomalies” are not a problem: DW assumes infrequent and off-line updates.

But, if you want to avoid the redundancy, go to *snowflake schema*.

■

Snowflake Schema

Use one table *for every level* of every dimension. Note foreign keys “KindID” in Students.

Students			Kinds		Facts		
sID	Student	KindID	KindID	Kind	cID	sID	Grade
0	Jill	0	0	grad	0	0	87
1	Jack	0			0	2	73
2	Al	1	1	u/g	1	1	25
					1	2	95
					2	1	89
					2	2	62

Course tables are on the next slide...

Snowflake Example: Courses

Note foreign keys “DiscID” from Courses into Disciplines, and “fID” from Disciplines into Faculties.

Courses			Disciplines			Faculties	
cID	Course	DiscID	DiscID	Discipline	fID	fID	Faculty
0	CS5678	0	0	CSCourses	0	0	Science
1	EE2222	1					
2	CS4002	0	1	EECourses	1	1	Engg

Star’s Courses table can be obtained by

$$\pi_{cID,Discipline,Faculty}(Courses \bowtie Disciplines \bowtie Faculties)$$

Level Indicator

Berson and Smith[BS97] talk about a “Level Indicator”, allowing data at different granularity to be stored in one table.

Probably a bad idea: invitation to mess up your query!■

Example:

Course	C Level	Student	S Level	Grade
CS5678	1	Jill	1	87
...
CSCourses	2	grad	2	??

How might the level indicator arise?

Perhaps the DW tries to mix aggregated facts with base-data facts. \Rightarrow the DW designer is “at fault”. ■

Or, perhaps the data is naturally available at different “granularity”. ■

Eg, before 1979, company reported sales only by the week. After 1979, sales reported daily. Do we really want to lose “by day” info for 1980-now? ■ Or give up pre-1980 sales data?

Avoiding the level indicator

Basic idea: sweep all facts with same levels into their own table.

We could keep computed aggregates as (maybe materialized) views.

Snowflake scheme is handy here.

Avoiding, cont.

Fact-1-1

cID	sID	Grade
0	0	87
...

Fact-2-2

DiscID	Kind	Grade
0	0	??
...

Nevertheless, many recommend avoiding the Snowflake.

References

- [BS97] Alex Berson and Stephen J. Smith. *Data Warehousing, Data Mining, and OLAP*. McGraw-Hill, 1997.
- [C⁺01] Surajit Chaudhuri et al. Database technology for decision support systems. *IEEE Computer*, pages 48–55, December 2001.