

---

---

# Decision Support and OLAP

30 May 2010

Lecture 11

---

# Topics for Today

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - CUBE
  - ROLLUP

Source: R&G 25.1-25.4

# Moving to Decision Support

---

Main goal of databases so far: **Support lots of fast transactions over a set of tables**

- Selects, updates, deletes, and comparisons

Called *Online Transaction Processing*

---

The nineties brought: **Decision Support**

- Less concern on: doing lots of fast updates, reads, and writes
- **More concern on:** generating **summarized data** in a **flexible** and **ad-hoc** way
- Meaning: **anybody** should be able to produce complex and readable summaries

Delivering this required **two developments**:

- **Visual applications** to let users visually design queries
- **DBMS support** to do computing and summarizing in the database

# OLAP and BI

---

We'll talk about [Decision Support](#) and [Online Analytic Processing \(OLAP\)](#)

- The techniques are often rolled up as [business intelligence](#), another 1990s buzzword
  - It implies using new [data analysis technologies](#) to extract [meaning](#), [patterns](#), and [insights](#) from [vast quantities](#) of data

Microsoft is the [biggest player](#) in the OLAP market, but it sells its OLAP extensions separately from the SQL Server package (so we don't have them in Kinneret)

[MySQL](#) offers a small amount of support - a `WITH ROLLUP` query - which we will see later

Most of the techniques and functions at the database level can be rewritten [by hand](#) with regular SQL

- The interesting value-added services from OLAP and BI are the custom applications usually sold with them

# What is Decision Support?

---

**Decision support:** the ability of a database or DB application to help organizations get “big picture” information via historical analysis and data cross-referencing

---

**How did it become important?**

- Relational DBMS technology is mature and getting cheaper
- Data storage and bandwidth is extremely cheap (per gigabyte/terabyte)
- Most organizations do electronic transactions, have electronic records, and communicate electronically

**Therefore:**

- Organizations began keeping huge masses of information (gigabytes, terabytes) on those cheap disks and DBs
- The quantity of data stored exceeds the ability for any one person to understand it
- Organizational leaders like to make decisions based on scientific (numerical or quantitative) analysis

**That lead to:**

- A demand for tools to give analytical summaries and projections based on huge amounts of data
- The DB vendors began to make tools to meet the demand

# Decision Support Tasks

---

## Recall Poria hospital which we visited last year:

- Each department of the hospital has its own data to store, but all of it is collocated in the same server room
- The servers are logically divided between the departments, but applications can cross over the boundaries to grab information from different places
- The hospital IT head wanted to introduce business intelligence analysis in his systems

---

## That means:

- The ability to see information about a patient's entire history of treatment in the hospital (across time and departments)
- Generate reports about the number of patients treated, their needs, outcomes, and costs
- Enable doctors in any department to see all relevant data about their patients
- Enable hospital administrators to receive reports about the hospital's performance.

# Decision Support Needs

---

**Decision support queries** generally have the following attributes:

- They summarize **large quantities** of data from many tables
- They include **many correlation checks** (WHERE) across time and space (year/city, quarter/store number)
- They pay special attention to **time** (week, month, quarter, year)
- They try to find **trends** and discover **patterns**

These are commonly called **OLAP queries** (they're next)

---

**Decision support applications** have special data needs:

- Data must be available on **one site** for analysis
  - Not be bogged down in network communication latencies
- Data needs to be **organized in a uniform manner**
- Data needs to be **cleaned and filtered**
  - no empty fields, incomplete data
- Data **does not need to be the most recently** available
  - 1 week, 1 month, 1 quarter, or (even) 1 year old is **ok**
- Data **does not need to be fully detailed** – summaries are ok

Such a database is called a **Data Warehouse** (we'll talk about them later)

---

# What's missing in SQL92?

---

Standard SQL92 falls short in dealing with the above needs:

1. WHERE checks are **slow** to run when there are lots of **OR conditions** in them
  - This is a weakness of SQL92 implementations which is bothersome for massive queries
  - Why?
2. Many **statistical functions** are not included in the standard SQL92 language so they must be written at the **application level**
  - Example: Median, Mode, Standard Deviation, Regression, Fitting
3. SQL lacks good support for **time** as an abstraction
  - Comparing, bucket creation, truncating
  - Some vendors added it afterwards
4. Many reports require many **related queries** which end up producing a single result – a report.
  - Most DBMS' can't recognize savings which can be achieved by saving or combining outputs from various related queries

# Decision Support Tools

---

Three general types of tools are on the market:

1. **Online Analytic Processing (OLAP)** databases and DB tools which offer a **wider form** of SQL including more **advanced grouping and filtering** options
2. Regular, commercial SQL DBMS' which offer **some extensions** for optimizing OLAP style queries
  - MS SQL Server and Oracle offer some features in their regular products
3. **Exploratory data analysis tools** (or **data mining tools**) which help experts discover trends and patterns that they weren't aware of before
  - Less about complex prepared queries and more about simple, but unexpected patterns
  - **There's a separate course on this**

# So Far

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - CUBE
  - ROLLUP

# OLAP: Multidimensional Data Model

In OLAP queries we often think of the data as residing in a **multidimensional array**

- Instead of the standard relation-table based model
- So queries have a bit of a **different semantics** than what we have seen until now

In the multidimensional model, we focus on a particular **measure** which is a numeric value

- The measure depends on a set of **dimensions** which we use to **categorize** and **understand** the measure data

Example: In a sales-oriented database:

- **Sales** is the measure
- **Product** (pid), **Location** (locid), and **Time** (timeid) are dimensions

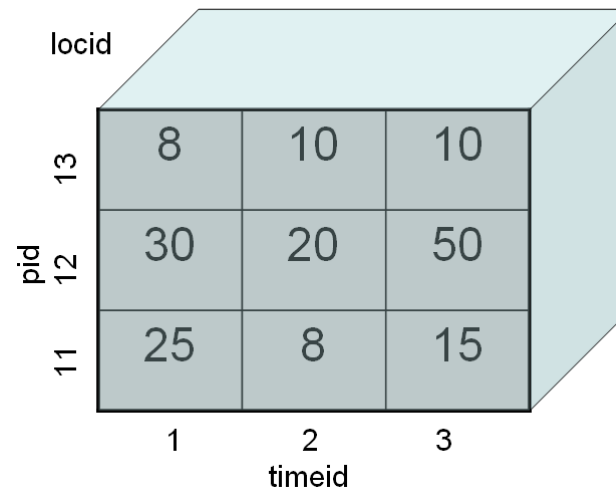
locid	timeid	pid	Value
11	1	11	25
11	2	11	8
11	3	11	15
12	1	11	30
12	2	11	20
12	3	11	50
13	1	11	8
13	2	11	10
13	3	11	10

# Storage?

---

How do we store the multidimensional array?

1. Could do it in normal relations
  - Then the “multidimensional array” is just imaginary
  - This is called a **Relational OLAP (ROLAP) system**
2. Could do it in a real multidimensional array/relations
  - This is called a **Multidimensional OLAP (MOLAP) system**



# ROLAP Example

**Example:** The locations and product relations relate the dimensions to the **measure** (sales)

- The sales table is called the **fact table**

Time Dimension

timeid	Date	Week	Month	Quarter	Year	Holiday?
1	1 Jan 10	1	1	1	2010	T
2	1 April 10	13	4	2	2010	F
3	1 July 10	26	7	3	2010	F

Products Dimension

pid	pname	category	price
11	Lee James	Apparel	25
12	Zork	Games	18
13	Biro Pen	Stationery	2

Locations Dimension

locid	city	state	country
1	Madison	WI	USA
2	Fresno	CA	USA
3	Chennai	TN	India

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26
12	2	2	45
12	3	2	20
13	1	2	20
13	2	2	40
13	3	2	5

# Dimensions

---

---

Dimensions are built of attributes which normally represent different levels of abstraction

- The schema for the dimensions is:

*Locations*(*locid*:integer, *city*:string, *state*:string, *country*:string)

*Products*(*pid*:integer, *pname*:string, *category*:string, *price*:real)

*Times*(*timeid*:integer, *date*:string, *week*:integer, *month*:integer, *quarter*:integer, *year*:integer, *holiday\_flag*:boolean)

The attributes of the dimensions are in a lattice structure, so:

- We can perform queries at different levels of specificity
- The tables aren't (necessarily) normalized

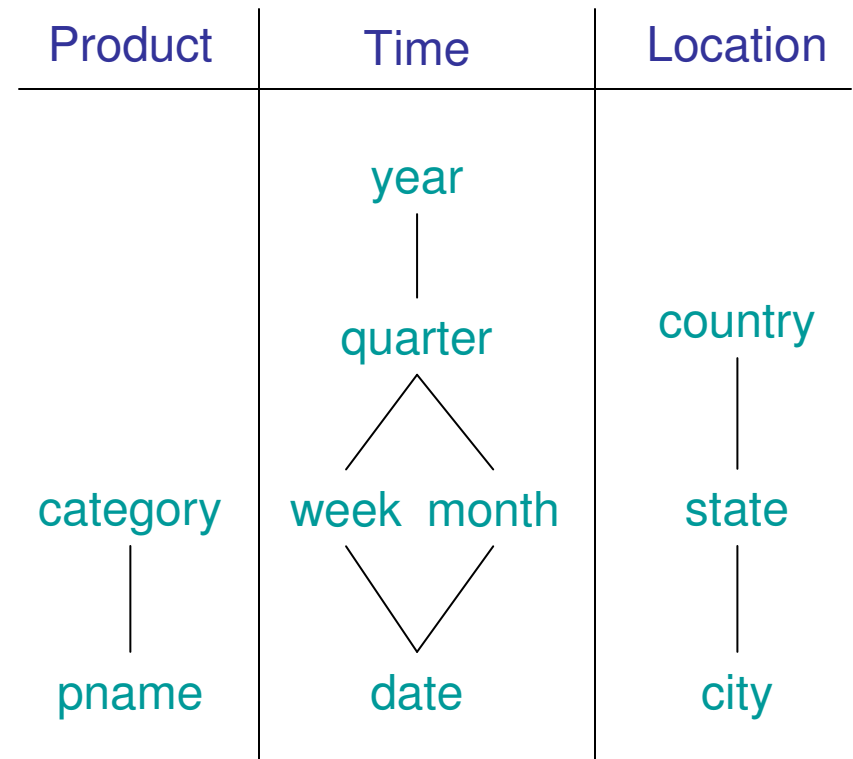
# Dimensions Lattice

A **lattice** shows the hierarchical relationship between the attributes

More about time:

1. Week and month aren't parent-child since a week may **cross multiple months**
2. Time is **very detailed** since the flags are necessary for business analysis
  - DateTime is **too generic**, we may need quarters and holiday information

**Note:** The DBMS may come with functions to extract the **quarter**, **month**, **week**, from the date, so we **might not** need to write this new



# So far

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - ROLLUP
  - CUBE
- Window Queries
  - PARTITION BY

# Multidimensional Database Design

A schematic of the database schema discussed:

Products

pid	pname	category	price
-----	-------	----------	-------

Locations

locid	city	state	country
-------	------	-------	---------

Sales

pid	timeid	locid	sales
-----	--------	-------	-------

Times

timeid	date	week	month	quarter	year	holiday_flag
--------	------	------	-------	---------	------	--------------

# Star Schema

---

This is called a **Star Schema** since it has a central relation (Sales) which is connected to all of the dimensions via foreign keys

The star schema is **very common** to find in OLAP databases  
- it relates the fact table to the dimensions tables

The fact table is generally designed to be **very tight** - only ids (potentially system generated) to the dimensions and columns of data

- The fact table is normally **highly normalized** - BCNF in this case

# Dimensions

---

Dimension tables are **not normalized**. Why?

- Usually they are **very small** compared to the size of the fact table, so having redundancy isn't so bad
- Usually they are **static** and so don't worry about update/delete/modification anomalies
- They are used **heavily in joins**, so they are kept together to save runtime on doing lots of extra joins

# What the DBMS Stores

---

An OLAP DBMS will store the fact table along with **summary tables** which are prepared summaries of important data

OLAP queries may use the summarized tables for faster results when **statistics** are requested

- Deciding **which tables to summarize and how** is an important part of MOLAP/ROLAP design

Some storage and indexing techniques available to help MOLAPs are in Section 25.6

- Read it on your own if you're interested.

# So far

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - CUBE
  - ROLLUP

# Multidimensional Aggregation Queries

---

---

OLAP is meant to support non-SQL experts, so the queries are designed to be easier to understand

- More like a spreadsheet (Excel)

Queries will show either a (1) new presentation of the data or (2) a summary

- Let's see an example and then how it's written

# Rollups

---

**Single Dimension Rollup:** Summaries based on one dimension

**Example:** Find the total sales, find the sales for each city, find the sales for each state

Total Sales	Fresno	176	CA	223
399	Madison	223	WI	176

The outcome depends only on one dimension, the other two are summarized away

Another option is to summarize of data at different levels of the hierarchy

- Called a *roll-up table*

# Cross-Tabulation Example

Area	Store Code	Store Name from Sales	Other Income	Regular Income	Total Income	Expenses	Gross Profit from Income	Expense %
North	12	Kochav Hatzafon	1,250.00	233.00	1,483.00	(560.00)	923.00	38%
	14	Hamazliach	950.00	145.00	1,095.00	(349.00)	746.00	32%
	17	Hashachar	1,145.00	350.00	1,495.00	(443.00)	1,052.00	30%
	25	Hakrayot	950.00	120.00	1,070.00	(356.00)	714.00	33%
	29	Mikol Tuv	1,355.00	234.00	1,589.00	(754.00)	835.00	47%
		<b>Area Total</b>	<b>5,650.00</b>	<b>1,082.00</b>	<b>6,732.00</b>	<b>(2462.00)</b>	<b>4,270.00</b>	<b>37%</b>
Center	4	Mercaz Hamichirof	1,567.00	452.00	2,019.00	(763.00)	1,256.00	38%
	13	Hachanut Hamercazit	1,789.00	641.00	2,430.00	(952.00)	1,478.00	39%
	19	Tov Li	2,341.00	437.00	2,778.00	(865.00)	1,913.00	31%
	24	Lev Hareshet	2,567.00	539.00	3,106.00	(1043.00)	2,063.00	34%
	27	Tov Taam	3,057.00	768.00	3,825.00	(1253.00)	2,572.00	33%
	36	Mercaz Hayokra	2,376.00	350.00	2,726.00	(983.00)	1,743.00	36%
	38	Tov Vizol	2,184.00	237.00	2,421.00	(668.00)	1,753.00	28%
		<b>Area Total</b>	<b>15,881.00</b>	<b>3,424.00</b>	<b>19,305.00</b>	<b>(6527.00)</b>	<b>12,778.00</b>	<b>34%</b>
South	5	Lev Haezro	1,563.00	340.00	1,903.00	(756.00)	1,147.00	40%
	9	Tiv Lakol	1,769.00	255.00	2,024.00	(458.00)	1,566.00	23%
	22	Mercaz Hanegev	2,546.00	675.00	3,221.00	(665.00)	2,556.00	21%
	45	Mifratz Hadromi	2,174.00	770.00	2,944.00	(897.00)	2,047.00	30%
		<b>Area Total</b>	<b>8,052.00</b>	<b>2,040.00</b>	<b>10,092.00</b>	<b>(2776.00)</b>	<b>7,316.00</b>	<b>28%</b>
		<b>Total</b>	<b>29,583.00</b>	<b>6,546.00</b>	<b>36,129.00</b>	<b>(11765.00)</b>	<b>24,364.00</b>	<b>33%</b>

# Drilling Down

---

A roll up table can be made more detailed on request on a given dimension

- Called a **drill down**

It's common to show the summary table and enable drill down by a click on the table

- A drill down on the previous table could show the details of a specific location's sales by Product

# Pivot Tables

---

Another common tool is **pivoting**: Creating summary tables based on one or two dimensions

In a pivot table, each value in the dimension is a row/column and its values in the fact table are detailed and then summed

- The result is a [cross-tabulation](#)
- 

**Example:** A pivot of the sales fact table by the location and time dimensions

- The location dimension is rolled up to the state attribute
- The time dimension is rolled up to the month attribute

	WI	CA	Total
Jan	63	81	144
April	38	107	145
July	75	35	110
Total	176	223	399

# Slicing and Dicing

---

Some more buzzwords:

- Creating tables based on an equality predicate for a particular dimension — **Slicing**
- Range selection over dimensions — **Dicing**
- Come from imagining how we would be cutting up a cube of data
- MS SQL even calls the OLAP multidimensional data sources “**Cubes**”

# A Note on Statistical Databases

---

Many of the operations above are similar to operations performed in **statistical databases**

A **statistical database (SDB)** is designed to produce only **statistical summaries** of the data that it contains

- Usually for privacy reasons since the raw data may be too sensitive to release to the public or to researchers

The application area of SDBs differs from OLAP in that SDBs place an inordinate amount of focus on protecting the anonymity and privacy of the data presented

- For example, if a query would result in a summary based on less than size  $k$  rows, the query **wouldn't even be allowed**

OLAP in contrast is concerned **solely with the efficient presentation of summaries of vast quantities of data**

- The summarization tools are essential, but there is not focus on the protection of data or on obfuscating individual records

# So Far

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - ROLLUP
  - CUBE
- Window Queries
  - PARTITION BY

# Pivot Table Example

---

In order to get the table shown, we need to run a bunch of queries:

```
SELECT T.month, L.state, SUM (S.sales)
FROM Sales S, Times T, Locations L
WHERE S.timeid = T.timeid AND S.locid = L.locid
GROUP BY T.year, L.state
```

```
SELECT T.month, SUM(S.sales)
FROM Sales S, Times T
WHERE T.timeid = S.timeid
GROUP BY T.month
```

```
SELECT L.state, SUM(S.sales)
FROM Locations L, Sales S
WHERE L.locid = S.locid
GROUP BY L.state
```

```
SELECT SUM(S.sales)
FROM Sales S, Locations L
WHERE S.locid = L.locid
```

	WI	CA	Total
Jan	63	81	144
April	38	107	145
July	75	35	110
Total	176	223	399

# How many pivots?

---

To compute a pivot table on **two dimensions** and we need 4 queries

- We could choose any two dimensions for the pivot table, so with  $k$  dimensions there are  $2^k$  possible SQL queries to write
- We can't pre-compute all of them, but ones that are **commonly used we can and should**
  - This goes back to what we mentioned before about design

For ones we **don't pre-compute**, we can speed them up:

- Since the queries are **related**, the DBMS can run them all together to save time in the calculations
- This also saves the user the time of typing all of them

---

To do pivots easier, SQL:1999 introduced **two new operators** which modify the GROUP BY commands:

- **CUBE**
- **ROLLUP**

# CUBE Command Example

The CUBE operation is identical to running a GROUP BY separately on all subsets of the dimensions listed

Example query:

```
SELECT T.month, L.state,
SUM(S.sales)
FROM Sales S, Times T,
Locations L
WHERE S.timeid = T.timeid
AND S.locid = L.locid
GROUP BY CUBE (T.month,
L.state)
```

The query generates the table:

The *nulls* indicate where the summations for the dimension are

T.month	L.state	SUM(S.sales)
Jan	WI	63
Jan	CA	81
Jan	<i>null</i>	144
April	WI	38
April	CA	107
April	<i>null</i>	145
July	WI	75
July	CA	35
July	<i>null</i>	110
<i>null</i>	WI	176
<i>null</i>	CA	223
<i>null</i>	<i>null</i>	399

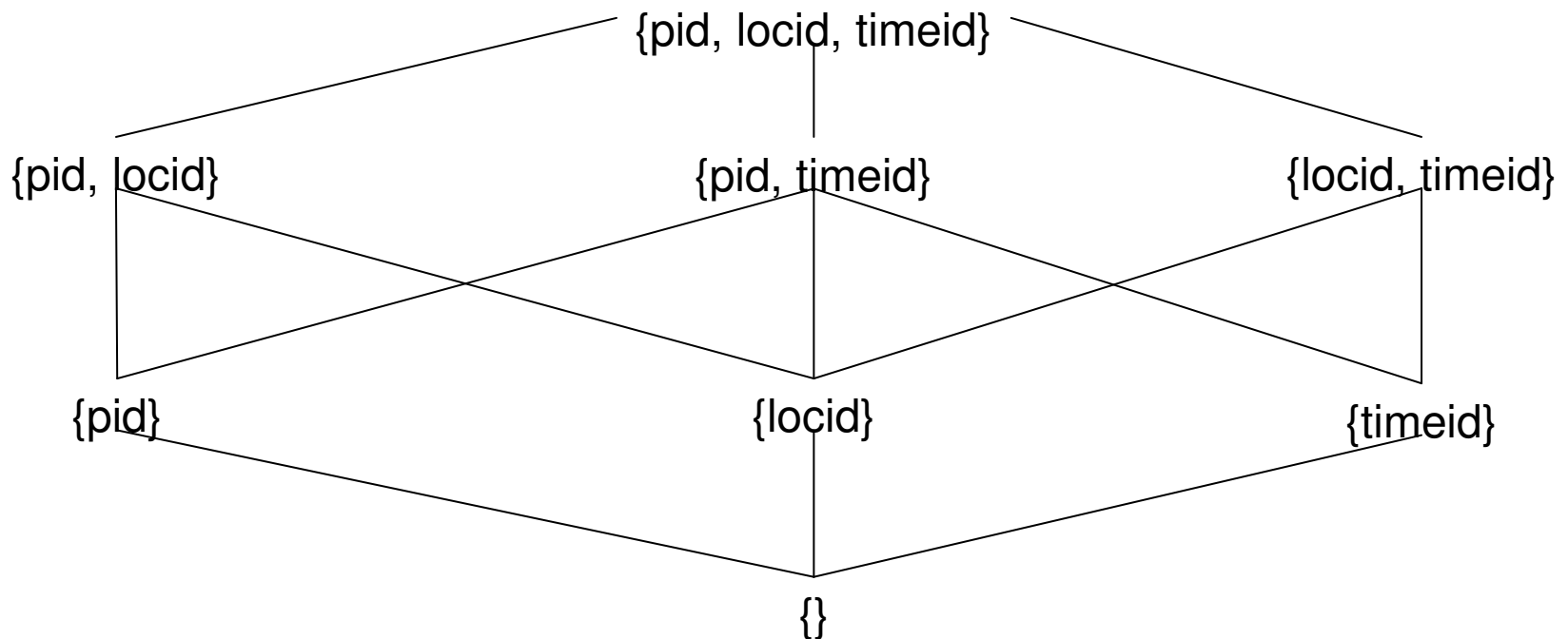
# CUBE Command

---

The CUBE calculates the powerset of all dimensions

- Running CUBE on more than two dimensions gives us a lot more summations
- For  $k$  dimensions, we have about  $2^k$  subsets

**Example:** `SELECT SUM(S.Sales) FROM Sales S GROUP BY CUBE (pid, locid, timeid)`



# ROLLUP

ROLLUP is similar (also modifies GROUP BY), but it doesn't compute all subsets

Example:

```
SELECT T.month, L.state,
SUM(S.sales)
FROM Sales S, Times T,
Locations L
WHERE S.timeid = T.timeid
AND S.locid = L.locid
GROUP BY ROLLUP (T.month,
L.state)
```

Computes all (year, state) values and per month sums

- **Not** for each state by itself (*null*, WI, 176), (*null*, CA, 22)

T.month	L.state	SUM(S.sales)
Jan	WI	63
Jan	CA	81
Jan	<i>null</i>	144
April	WI	38
April	CA	107
April	<i>null</i>	145
July	WI	75
July	CA	35
July	<i>null</i>	110
<i>null</i>	<i>null</i>	399

# Conclusion

---

- Decision support
- OLAP
- Multidimensional Database Design
- Multidimensional Aggregation Queries
  - CUBE
  - ROLLUP