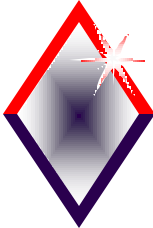


# *What is OLAP - On-line analytical processing*

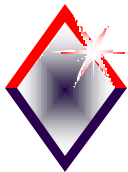
©Vladimir Estivill-Castro

School of Computing and Information Technology

With contributions for J. Han



1



## *Introduction*

- ◆ When a company has received/accumulated data, it often wants a report
  - ◆ to get a summary, to visualize, to make decisions
- ◆ This is often done with some IT tools
  - ◆ Mainframe systems (old and new)
  - ◆ SQL, ODBC, JDBC, etc



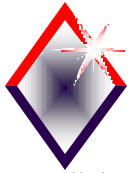
© Vladimir Estivill-Castro

2



## *Problems*

- ◆ Report design (making) can take a long time with traditional systems
  - ◆ does not facilitate explorative views on the data
  - ◆ with large data sets and tricky queries many tables may be involved (many locations)
- ◆ Changes in reports can require modifications in legacy applications



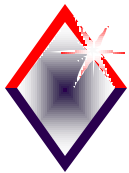
## *Data Warehousing*

- ◆ “ A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management’s decision-making process.” --- W. H. Inmon
- ◆ A data warehouse is
  - ◆ A decision support database that is maintained separately from the organization’s operational databases.
  - ◆ It integrates data from multiple heterogeneous sources to support the continuing need for structured and /or ad-hoc queries, analytical reporting, and decision support.



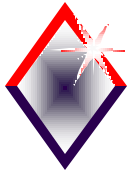
## *What is OLAP*

- ◆ OLAP: On-Line Analytic Processing
  - ◆ Starts with “summarizing” the data before it is possible to execute the queries (to receive a report)
    - ◆ this is building “the cube”
    - ◆ this can take a long time
    - ◆ both more efficient response for analysis queries
  - ◆ Data (summarization) is represented as cubes and subcubes



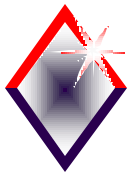
## *OLAP vs Data Mining*

- ◆ Data Mining: Finding patterns in data
- ◆ OLAP: reporting data, visualizing data, interaction with views of the data.



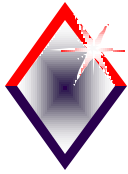
## Database terminology

- ◆ A *tuple* (data value) is a single data field in a database. It can be a date, a name, a number, etc.
- ◆ A *record* is a set of tuples. All tuples in a record are information about an entity
- ◆ A *table* is a set of records all from the same domain. Each *row* in a table is a unique record and each *column* is the specific tuple (or attribute)



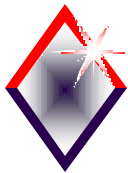
## Data Models

- ◆ Data is either *denormalized* or *normalized*
  - ◆ Denormalized: Multiple rows repeat the same information
  - ◆ Normalized: Only one row has the information
- ◆ Star Schema
  - ◆ Developed by R. Kimball
  - ◆ A denormalized approach
  - ◆ Starts with a central fact table that corresponds to facts about a business

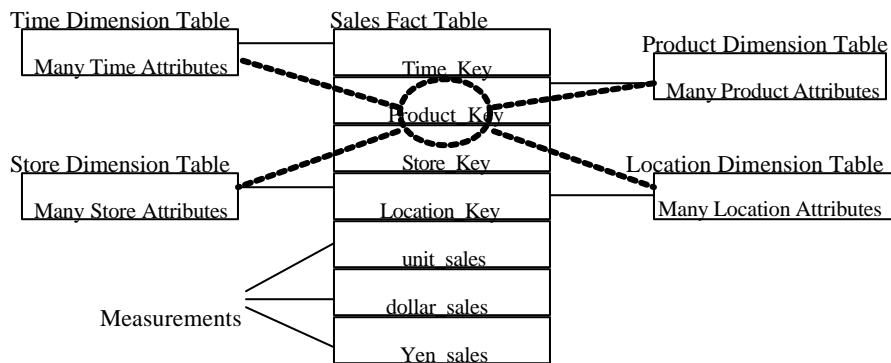


## Central Fact Table

- ◆ Facts about the business
  - ◆ Each row contains a combination of facts that makes it unique
  - ◆ The keys to make it unique are called *dimensions*
  - ◆ Each dimension is associated with a *dimension table* that contains information specific to the dimension.

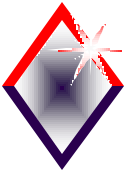
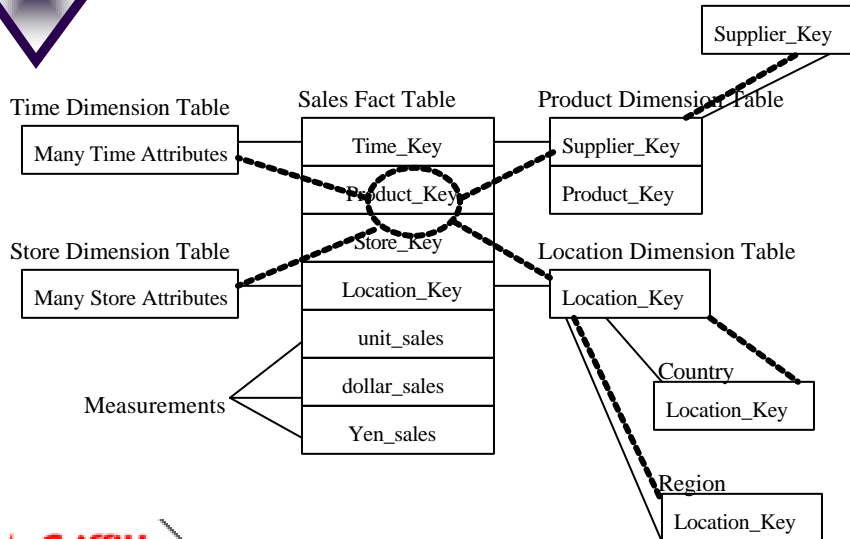


## Example of Star Schema

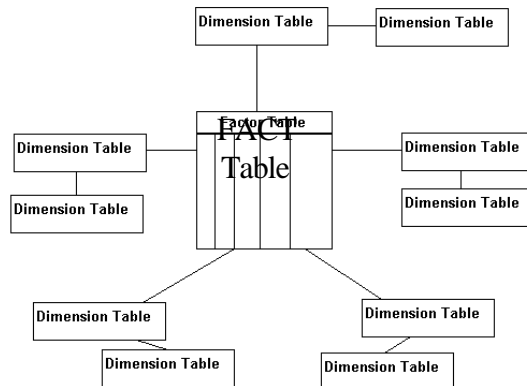




## Example of a Snowflake Schema

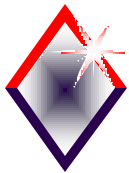
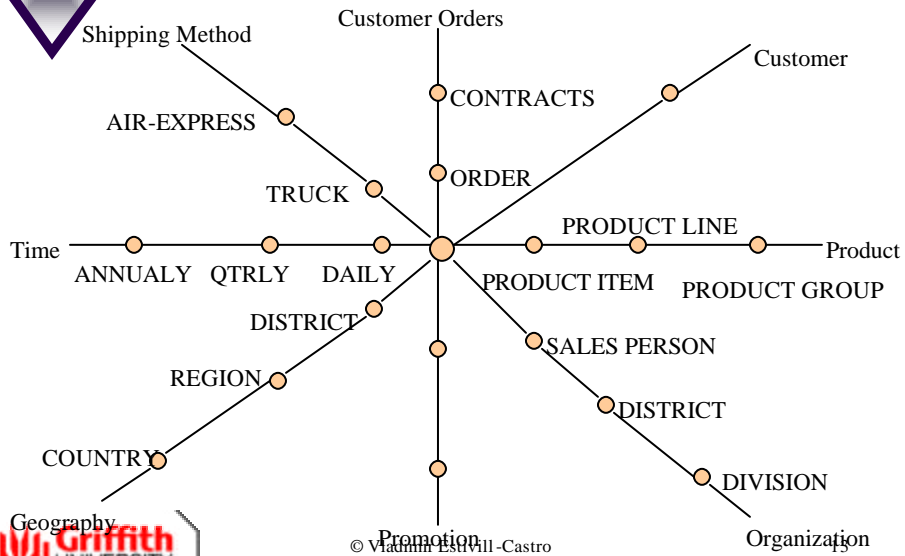


## General Structure

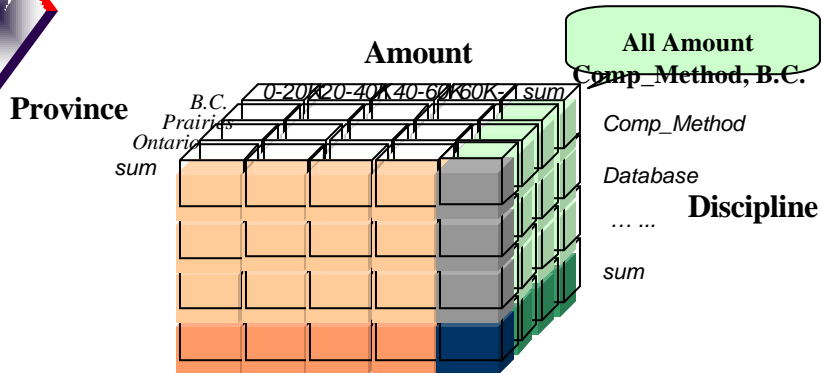




## A Star-Net Query Model



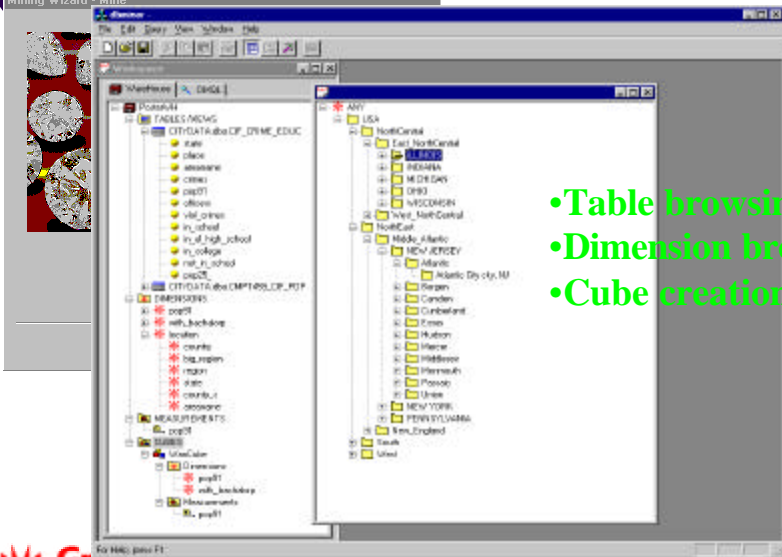
## Construction of Data Cubes



- Each dimension contains a hierarchy of values for one attribute
- A cube cell stores aggregate values, e.g., count, sum, max, etc.
- A “sum” cell stores dimension summation values.
- Sparse-cube technology and MOLAP/ROLAP integration.
- “Chunk”-based multi-way aggregation and single-pass computation.



## View of Warehouse & Hierarchies

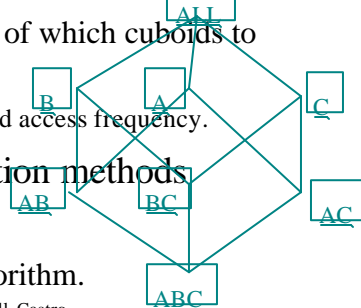


The screenshot shows a software interface with two panes. The left pane displays a tree view of tables and dimensions, including categories like 'TABLES:MEAS', 'DIMENSIONS', and 'MEASUREMENTS'. The right pane shows a hierarchical view of geographical regions, such as 'USA', 'NorthCentral', 'East\_NorthCentral', 'Midwest', and 'South'. A red diamond icon with a starburst is positioned in the top-left corner of the slide.

- Table browsing
- Dimension browsing
- Cube creation

## Efficient Data Cube Computation Methods

- ◆ Data cube can be viewed as a lattice of cuboids
  - ◆ The bottom-most cuboid is the base cube.
  - ◆ The top most cuboid contains only one cell.
- ◆ Materialization of data cube
  - ◆ Materialize every (cuboid), none, or some.
  - ◆ Algorithms for selection of which cuboids to materialize.
    - ◆ Based on size, sharing, and access frequency.
- ◆ Efficient cube computation methods
  - ◆ ROLAP algorithms.
  - ◆ Array-based cubing algorithm.

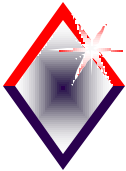






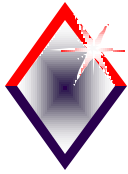
## *OLAP: On-Line Analytical Processing*

- ◆ A multidimensional, LOGICAL view of the data.
- ◆ Interactive analysis of the data: drill, pivot, slice\_dice, filter.
- ◆ Summarization and aggregations at every dimension intersection.
- ◆ Retrieval and display of data in 2-D or 3-D crosstabs, charts, and graphs, with easy pivoting of the axes.
- ◆ Analytical modeling: deriving ratios, variance, etc. and involving measurements or numerical data across many dimensions.
- ◆ Forecasting, trend analysis, and statistical analysis.
- ◆ Requirement: Quick response to OLAP queries.



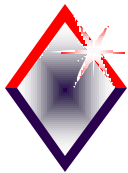
## *OLAP Architecture*

- ◆ Logical architecture:
  - ◆ OLAP view: multidimensional and logic presentation of the data in the data warehouse/mart to the business user.
  - ◆ Data store technology: The technology options of how and where the data is stored.
- ◆ Three services components:
  - ◆ data store services
  - ◆ OLAP services, and
  - ◆ user presentation services.
- ◆ Two data store architectures:
  - ◆ Multidimensional data store: (MOLAP).
  - ◆ Relational data store: Relational OLAP (ROLAP).



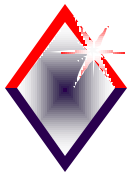
## *Cubes for representing data*

- ◆ OLAP considers two types of columns in denormalized data:
  - ◆ Dimensional columns
    - ◆ Contain information used for summarization
    - ◆ Take a fixed number of values (categorical)
    - ◆ Often its value is part of a hierarchy
      - ◆ location-code, postal code, state, region
  - ◆ Aggregate columns
    - ◆ Calculated amounts like counts, averages and sums



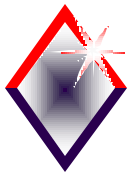
## *Design of a cube*

- ◆ Deciding which columns will be designated as dimensions and which will be designated as aggregates



## *An example database*

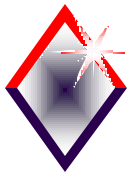
Name	Gender	Age	Source	Movie
Amy	F	27	Oberlin	Independence Day
Andrew	M	25	Oberlin	12 Monkeys
Andy	M	34	Oberlin	The Birdcage



## *Examples of questions (on-line queries)*

- ◆ What are the number of people and their ages by source?

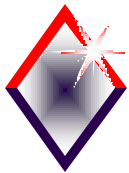
Source	Number	Average Age
1	103	31.41
2	23	39.35
3	54	35.04
4	28	33.43



## *Examples of questions (on-line queries)*

- ◆ What are the number of people from the two most populated sources by gender?

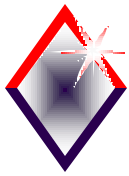
Source	Gender	Count
1	Female	55
1	Male	48
2	Female	16
2	Male	17



## *More examples*

- ◆ For what movies is the average age of the viewers over 35?

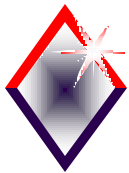
Movie Id	Average Age
110	50.00
48	46.00
30	46.00
23	45.13
25	44.80
107	44.00



## *More examples*

- ◆ The number of times each movie was seen for movies seen more than five times

Movie Id	Average Age
1	34
13	14
26	12
60	11
32	10
22	9



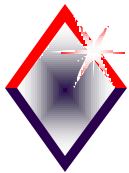
## *Moviegoers database*

- ◆ There are 3 candidates for dimensions
  - ◆ the movie
  - ◆ the gender of movie goers
  - ◆ the source of information (branch)
- ◆ There are two candidates for aggregations
  - ◆ the number of times that each movie was seen
  - ◆ the average age of the moviegoers

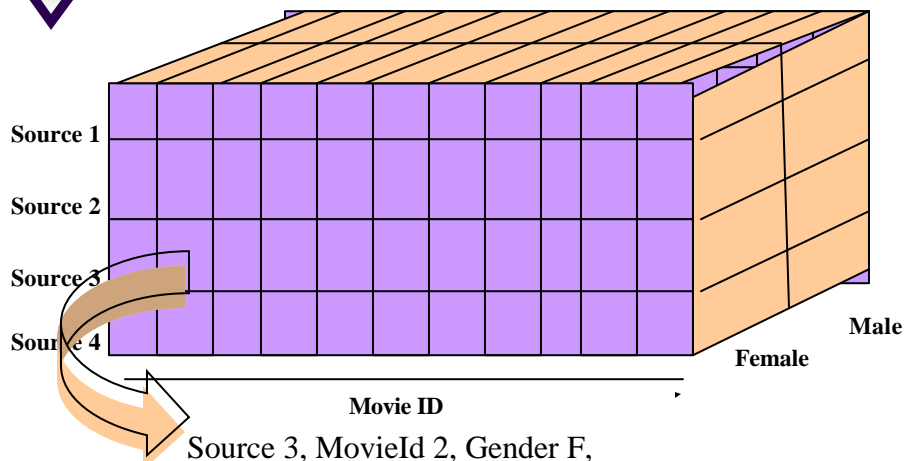


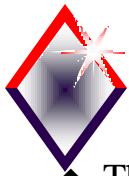
## *The moviegoers into a cube*

- ◆ Each dimensions corresponds to an axis in the cube
  - ◆ One dimension is the gender which split the axis into half since there are only two types of genders
  - ◆ The source of information is split into four parts since there are four different sources
- ◆ The cube contains  $S$  cells where
  - ◆  $S = \#ofSources \times \#ofGender \times \#of MovieIds$



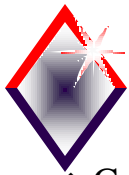
## *The Moviegoers cube*





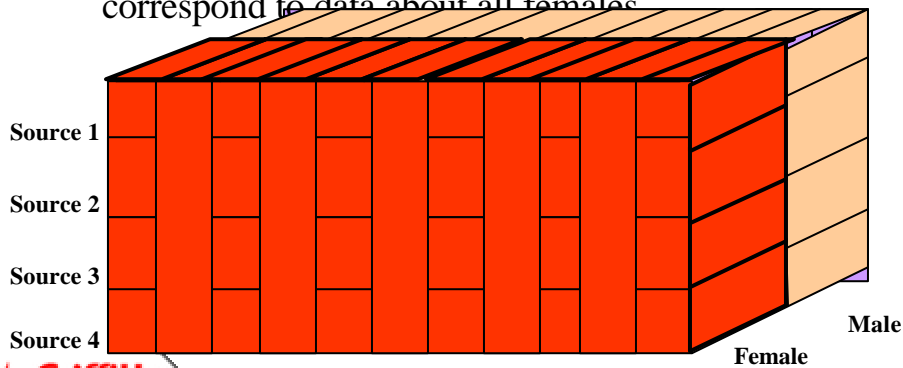
## Notes on cubes

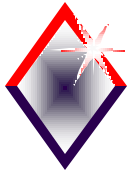
- ◆ The number of subcubes (cells) will not change unless the number of movies, genders or sources changes
  - ◆ This makes it possible to have an unlimited number of people in the cube
- ◆ Each cell contains aggregate information
  - ◆ The cell key is its coordinates
    - ◆ movie\_id, source, gender
  - ◆ The aggregate information are statistics
    - ◆ The sum of the ages, the number of rows with given key



## Notes on cubes

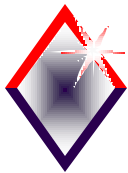
- ◆ Cubes have natural subcubes
  - ◆ All the front cells for a sub-cube that correspond to data about all females





## *The Cube Data Model*

- ◆ Each record must land in only one cell.
- ◆ The Data Model varies when attributes are numerical
  - ◆ continues values
- ◆ There is an assumption about hierarchical dimensions
  - ◆ Movies: action, comedy, drama
- ◆ Problems with dimensions that span multiple fields.



## *Core Operations of OLAP Systems:*

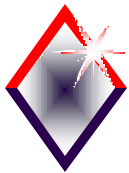
- ◆ *Rollup*: an aggregation on the data cube by either moving up the concept hierarchy or by the reduction of a dimension.
- ◆ *Drill Down*: the moving from the current data cube to a more detailed data cube by either adding a dimension, or moving down a concept hierarchy.
- ◆ *Slice*: this is where you select a dimension from the cube and display only it.
- ◆ *Dice*: creates a sub cube of the current cube by selecting one or more dimensions and the ranges for the values to be included.
- ◆ *Pivot*: this operation removes a dimension from a cube and replaces it with another.





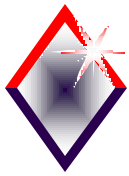
## *Continuous Values*

- ◆ They are clustered into ranges (for efficiency)
  - ◆ Ages grouped into
    - ◆  $0 < \text{young} \leq 22$  and  $22 < \text{old} < 100$
  - ◆ Even if age is chosen as a dimension it can still be used as an aggregate

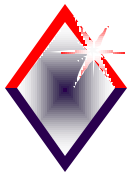
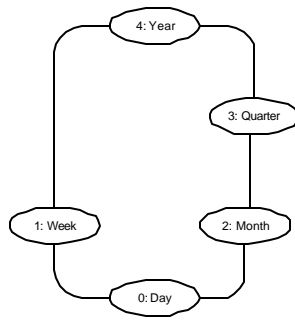


## *Hierarchical dimensions*

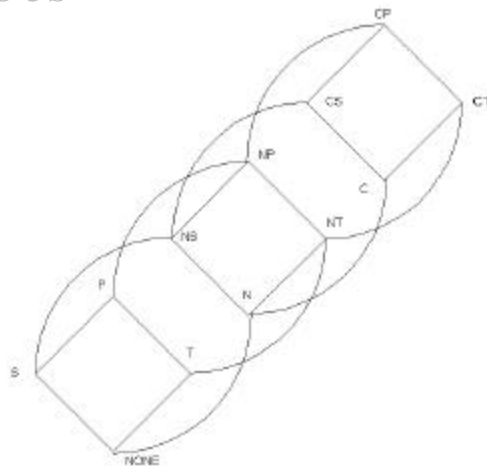
- ◆ A single dimension can some times seem appropriate for more dimensions than one
  - ◆ A date potentially represents information along several dimensions
    - ◆ day of the week, month, quarter and year
  - ◆ One solution is to use different dimensions
    - ◆ break data model and lots of redundancy
  - ◆ Second solution, organize into a hierarchy

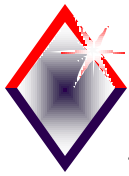


## Illustration of hierarchies



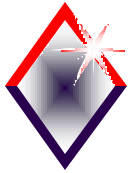
## Illustration of the lattice of cubes





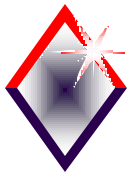
## *Dimensions that span multiple fields*

- ◆ If multiple columns correspond to a single dimension, preprocessing is required to merge into one dimension
  - ◆ If month, day and year data detail exists, the time dimension requires to consider these as one dimension
  - ◆ The preprocessing is guided by the interest of users.



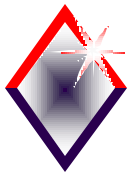
## *Storage architectures*

- ◆ ROLAP vs MOLAP
  - ◆ ROLAP (relational OLAP) stores the cube inside a RDBMS
    - ◆ takes advantage of many established features of the relational database (security, concurrent access, etc.)
  - ◆ MOLAP (multi-dimensional OLAP) stores the cube as multidimensional database (array) that is designed for the features and performance needs of OLAP



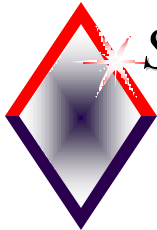
## *OLAP*

- ◆ Offer a powerful visualization tools
- ◆ It provides fast, interactive response times
- ◆ It is good for analyzing time series
- ◆ It can be useful to find clusters and outliers
- ◆ There are many vendors of OLAP products



## *OLAP*

- ◆ Setting up a cube can be difficult
- ◆ It does not handle continues values well
- ◆ Cubes can quickly become out of date
- ◆ It is not data mining
  - ◆ It may involve dangerous exploration of the data by users.

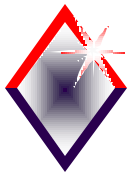


## *Selective Materialization:*

### An Effective Method for Spatial Data Cube Construction

*Jiawei Han, Nebosja Stefanovic and Krzysztof Koperski*

41



## *Selective Materialization*

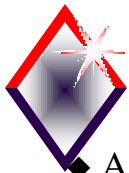
- ◆ Pre-introduction
- ◆ Introduction
- ◆ A model of spatial data warehouses
- ◆ Methods for Computing Spatial Measures in  
Spatial Data Cube Construction
- ◆ Performance Analysis
- ◆ Discussion



## *Pre-introduction*

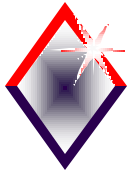
- ◆ Spatial data are the data related to objects that occupy space.
- ◆ A spatial database stores spatial objects represented by spatial data types and spatial relationships among such objects.

[<http://fas.sfu.ca/cs/people/GradStudents/koperski/personal/research/research.html>]



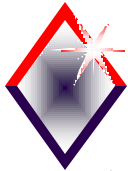
## *Introduction*

- ◆ A lot of systems collect a huge amount of spatial data
  - ◆ Satellite telemetry systems
  - ◆ Remote sensing systems
  - ◆ etc...
- ◆ We want to develop efficient methods for analysis and understanding of the data.
- ◆ In the paper, it is studied how to construct a *spatial data warehouse* and how to implement efficient Spatial OLAP (OLAP=familiar...)



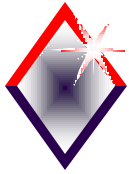
## *Introduction - Two examples*

- ◆ Example 1 - Regional weather pattern analysis
  - ◆ Over 3000 weather probes recording temperature and precipitation (rain, snow etc...) for a designated area.
  - ◆ A user may want to view weather patterns on a map by month, by region or maybe find a specific pattern by himself.
- ◆ Example 2 - Overlay of multiple thematic maps
  - ◆ A database stores different thematic maps in a database, such as maps of altitude, population and daily temperature.
  - ◆ A user may want to find relationships between population and altitude for example



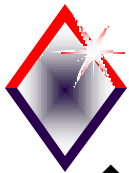
## *Two examples (Cont.)*

- ◆ To satisfy the desired user tasks, there are a couple of challenging issues to solve.
- ◆ The first challenge is to integrate all the data.
  - ◆ Data can be stored in different physical locations
  - ◆ Data can have different format
  - ◆ Data can be stored in databases from different vendors
  - ◆ Since this is implementation issues not related to the paper, it is assumed that the issues above are solved.



## *More challenges*

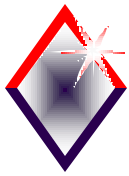
- ◆ The second challenge is the realisation of fast and flexible OLAP.
  - ◆ Different methods for storing and indexing spatial data for efficient storing and accessing has been studied intensively.
  - ◆ These methods cannot alone provide sufficient support for OLAP for spatial data since OLAP operations usually summarises data into dimensions with different levels of abstraction



## *A model of spatial data warehouses*

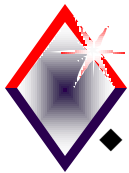
- ◆ A data warehouse is often designed for OLAP and usually adopts the star-schema model (central fact table and dimension tables).
- ◆ For a *spatial* data warehouse this model is usually a good choice as well.
- ◆ A spatial data cube can be constructed according to the dimensions and measures modelled in the warehouse.





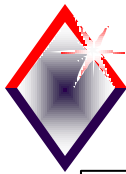
## *Three cases of modelling dimensions in a spatial data cube*

- ◆ Non-spatial dimension
  - ◆ From first example *temperature* and *precipitation* can be generalised as *hot* and *wet*
- ◆ Spatial to non-spatial dimension
  - ◆ Starts with a high level dimension that is spatial but after generalisation it becomes non-spatial. For example, *state* can be represented as spatial but can be generalised as *north\_west* or *big\_state*.
- ◆ Spatial to spatial dimension
  - ◆ Data that after generalisation still is spatial.

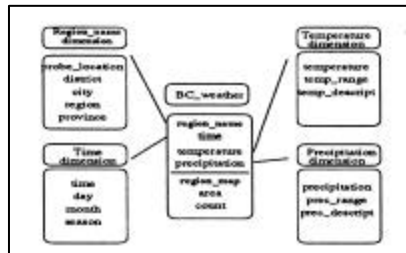


## *Modelling measures*

- ◆ A computed measure can be used as a dimension in a dimension (measure-folded)
- ◆ A spatial cube has two cases for modelling measures
  - ◆ Numerical measure - contains only numerical data
  - ◆ Spatial measure - contains one or many pointer(s) to spatial objects
    - ◆ If *temperature* and *precipitation* are grouped into one cell, then the spatial measure will contain pointers to the regions that satisfy those values.
- ◆ A non-spatial cube contains only non-spatial dimensions and numerical measures.



## Star modelling of example 1



Star model of a spatial DW

Hierarchy for temp. dimension

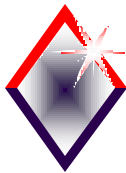
Temperature:  
any ⊃ (cold, mild, hot)  
cold ⊃ (below -20, -20 to -10, -10 to 0)  
mild ⊃ (0 to 10, 10 to 15, 15 to 20)  
hot ⊃ (20 to 25, 25 to 30, 30 to 35, above 35)

◆ Four dimensions:

- ◆ *temperature*
- ◆ *precipitation*
- ◆ *time*
- ◆ *region\_name*

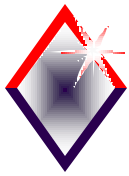
◆ Three measures

- ◆ *region\_map* (spatial)
- ◆ *area* (numerical)
- ◆ *count* (numerical)



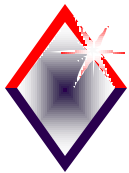
## How OLAP operations perform in a spatial data cube

- ◆ Slicing and dicing
  - ◆ Selects a portion of the cube based on the constant(s) in one or a few dimensions.
  - ◆ Can be done with regular queries
- ◆ Pivoting
  - ◆ Presents the measures in different cross-tabular layouts
  - ◆ Can be implemented in a similar way as in non-spatial cubes



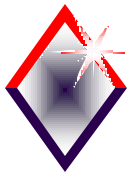
## *How OLAP operations perform in a spatial data cube (Cont.)*

- ◆ Roll-up
  - ◆ Generalises one or a few dimensions and performs appropriate aggregations in the corresponding measures
  - ◆ For non spatial measures aggregation is implemented in the same way as in non-spatial data cubes
  - ◆ For spatial measures, the aggregate takes a collection of spatial pointers
    - ◆ Used for map-overlay
    - ◆ Performs *spatial aggregation* operations such as region merge etc.



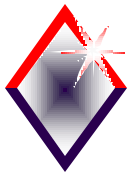
## *How OLAP operations perform in a spatial data cube (Cont.)*

- ◆ Drill-down
  - ◆ Specialises one or a few dimensions and presents low-level data
  - ◆ Can be viewed as a reverse operation of roll-up
  - ◆ Can be implemented by saving a low-level cube and performing a generalisation on it when necessary.
- ◆ Major implementation issues
  - ◆ Efficient construction of spatial cubes
  - ◆ Implementation of Roll-up and Drill-down operations



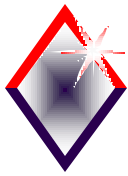
## *How OLAP performed in the example*

- ◆ Starts with a *Roll-up* on the time dimension from day to month
- ◆ After this, roll-up the temperature dimension
  - ◆ It is measure folded (continuous)
  - ◆ Start with calculating the average temperature grouped by month and by spatial region
  - ◆ Generalise the values to ranges such as cold, mild, warm...
- ◆ Do the same as above with the precipitation dimension



## *How OLAP performed in the example (Cont.)*

- ◆ The result of the roll-ups gives the following structure of the table
  - ◆ *Time* in month
  - ◆ *Temperature* in monthly average
  - ◆ *Precipitation* in monthly average
  - ◆ One spatial measure which is a collection of *spatial\_id*'s
- ◆ Here the dimension *region\_name* is dropped



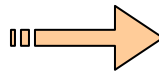
## Results from the roll-up

Region_sname	Time	Temperature	Precipitation
AA00	01/01/97	-4	1.6
AA01	01/01/97	-7	1.0
...	...	...	...
AA00	01/02/97	-4	2.5
AA01	01/02/97	-8	1.0
...	...	...	...

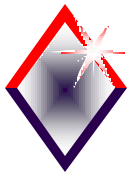


Time	Temperature	Precipitation	Collection of Spatial ids
March	cold	0.1 to 6.3	{AM4, AM3, ... X99}
March	cold	0.3 to 1.0	{AM10, A300, ... Y99}
...	...	...	...

Table roll-up

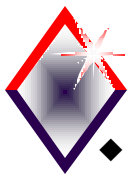


Generalise regions



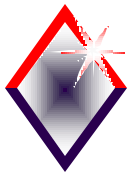
## Results from the roll-up (Cont.)

- ◆ Response time for the merging can only be acceptable if appropriate pre-computation is done
- ◆ Definitions:
  - ◆ A high-level view is called a cuboid
  - ◆ A pre-computed (and saved) cuboid is called a *materialised view* or a *computed cuboid*
- ◆ Some DW materialise every cuboid, some none, and some only a part of the cube (some cuboids).
  - ◆ Balancing is needed



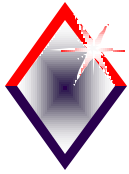
## *Methods for computing spatial measures in spatial data cube construction*

- ◆ There are (at least) three choices for computation of spatial measures
  - ◆ Collect and store the corresponding spatial object without pre-computation
    - ◆ They have to be computed on the fly
    - ◆ Good for cubes in view-only mode
  - ◆ Pre-compute and store rough approximations
    - ◆ Good for a rough view
    - ◆ If higher precision needed, compute on the fly
  - ◆ Selectively pre-compute spatial measures
    - ◆ Can require a large pre-computation
    - ◆ What to compute???



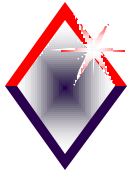
## *Methods for computing spatial measures in spatial data cube construction (Cont.)*

- ◆ Focus on how to select a group of mergeable spatial objects for pre-computation is needed
- ◆ Three factors to consider when judging whether materialisation should be done or not:
  - ◆ Potential access frequency of the generated cuboid
  - ◆ The size of the generated cuboid
  - ◆ How the materialisation of one cuboid may benefit computation of other cuboids



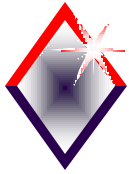
## *Methods for computing spatial measures in spatial data cube construction (Cont.)*

- ◆ There are two algorithms studied for this purpose
  - ◆ Pointer Intersection algorithm
  - ◆ Object Connection algorithm
- ◆ Both algorithms are similar in the way that
  - ◆ Given a set of cuboids associated with an estimated access frequency (*eaf*) and a minimum access frequency (*min\_freq*) threshold
  - ◆ A set of objects should be pre-computed if and only if  $eaf \geq maf$



## *The algorithms (Cont.)*

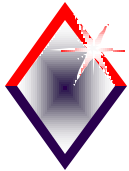
- ◆ The pointer intersection algorithm
  - ◆ First computes the intersections among the objects
  - ◆ Secondly it performs the (threshold) filtering and examines the object's corresponding spatial object's connections
- ◆ The object connection algorithm
  - ◆ Starts with examining the corresponding object's connections
  - ◆ At last it performs the threshold filtering



## The pointer intersection algorithm

```
(1) FOR cuboid_i = 1 TO max_cuboid DO
(2)   FOR cuboid_j = cuboid_i TO max_cuboid DO
(3)     FOR EACH cell_i IN cuboid_i DO
(4)       get_max_intersection(cell_i, cuboid_j, candidate_table);
(5)   frequency_computing &_filtering(candidate_table);
(6)   spatial_connectivity_testing(candidate_table, connected_obj_table);
(7)   shared_spatial_merging(connected_obj_table, merged_obj_table);
(8)   populate_cube(merged_obj_table);
```

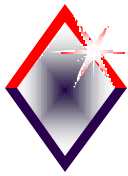
```
(1) PROCEDURE get_max_intersection(cell_i, cuboid_j, candidate_table) {
(2)   cell_j = get_first_cell(cuboid_j);
(3)   remaining_cell_i = cell_i;
(4)   WHILE (|remaining_cell_i| > 1 AND cell_j ≠ ∅) DO {
(5)     intersected_portion = get_max_intersect
      (remaining_cell_i, cell_j);
(6)     IF |intersected_portion| > 1
(7)     THEN insert_candidate(intersected_portion, candidate_table);
(8)     remaining_cell_i -= intersected_portion;
(9)     cell_j = get_next_cell(cuboid_j);
(10)  }
(11) }
```



## The object connection algorithm

```
(1) FOR cuboid_i = 1 TO max_cuboid DO
(2)   FOR cuboid_j = cuboid_i TO max_cuboid DO
(3)     FOR EACH cell_i IN cuboid_i DO
(4)       get_max_connected_intersection
      (cell_i, cuboid_j, candidate_connected_obj_table);
(5)   frequency_computing &_filtering(candidate_connected_obj_table);
(6)   shared_spatial_merging(candidate_connected_obj_table,
      merged_obj_table);
(7)   populate_cube(merged_obj_table);
```





## Performance analysis

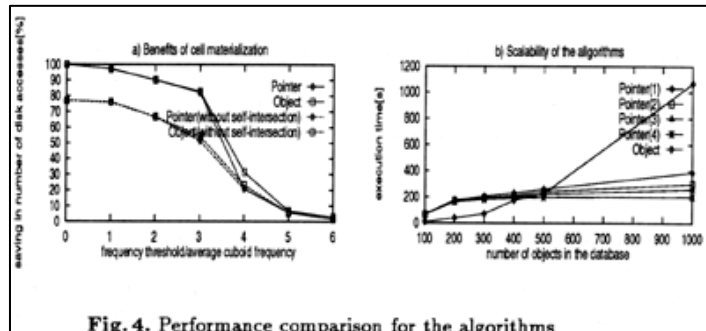
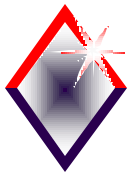


Fig. 4. Performance comparison for the algorithms



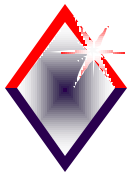
## Performance analysis (Cont.)

- ◆ The benefit of the materialised groups is studied
  - ◆ The number of pre-merged cuboids gets smaller with the increase of frequency threshold
  - ◆ Only a slight difference between effectiveness (between the two algorithms)
- ◆ The algorithms were tested with self- and non-self-intersection
  - ◆ With self-intersection, the benefit increased, but without the disk-accesses decreased



## *Performance analysis (Cont.)*

- ◆ Execution time was also examined (for pre-computation)
  - ◆ Maybe not as crucial as on-line running time, but it is concerned because the need to DW maintenance.
- ◆ For the object connection algorithm execution time was independent of the frequency threshold
  - ◆ Since frequency filtering is the last step in the algorithm
  - ◆ But, pointer frequency algorithm shows better performance when the frequency threshold increases due to fewer groups tested for spatial connectivity.



## *Discussion*

- ◆ What if the *eaf* does not exist
  - ◆ One solution is to assign an initial access frequency only to a level in the lattice (less work), based on assumption.
  - ◆ For example, assuming that medium level (county level in a province map) are accessed most frequently.
  - ◆ A frequency estimate can be adjusted based on later accessing records