Motivation

In business domains,
- Large amount of business data
- Data are in different formats and locations (heterogeneous vs. distributed)
- Decision makers need fast accesses of summarized information (usually) on a single site

Outline

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse design & architecture
- Data warehouse implementation
- From data warehousing to data mining

Data Warehouses

Loosely speaking, a data warehouse is
- a data store that integrates data from multiple heterogeneous sources to support decision making
- maintained separately from operational databases

A data warehouse system provides a solid platform of consolidated data for on-line analytical processing (OLAP) and data analysis

Data warehousing is a process of constructing and using data warehouses
Other Related terms

- **Data Warehouse**: enterprise-based
  - Concerns with decision subjects of the whole enterprise or organization
- **Data Mart**: department-based
  - Specialized single line of business warehouses e.g., within departments or groups of people

What is a data warehouse?

More precisely, a **data warehouse** (DW) is a
- subject-oriented
- integrated
- time-variant, and
- non-volatile (i.e., stable)
collection of data in support of management’s decision-making process [Inmon, 96]

Subject-oriented

A data warehouse:
- Is organized around **subjects of interest** e.g., customer, product, sales.
- Gives a simple and concise view of relevant issues by excluding non-useful data to the decision support
- Focuses on the modeling/analysis of data for decision makers, not on daily operations or transaction processing.

Integrated

- A data warehouse integrates multiple heterogeneous data sources: relational databases, on-line transaction records
- Moving (or converting) data to the warehouse requires
  - data cleaning and
  - data integration techniques
  to ensure consistency among different data sources in
  - naming conventions
  - encoding structures, attribute measures, etc.
  
  E.g., Hotel price: currency, tax, breakfast covered, etc.
Time-variant

Key structures in data warehouses
- contain “time” information explicitly or implicitly
- Not necessary in operational databases
- give historical perspective (e.g., past 5-10 years)
- Operational time period is shorter (e.g., current data)

Non-volatile

A data warehouse
- is a physically separate store of data transformed from the operational environment.
- does not require transaction processing, recovery, and concurrency control mechanisms
- requires only two operations in data accessing:
  - initial loading of data
  - access of data
  → Non-volatile

Data Warehouse vs. Heterogeneous DB

- Traditional heterogeneous DB integration: Query-driven
  - Builds wrappers/mediators on top of heterogeneous DBs
  → requires complex information filtering and integration processes
- Data Warehouse: Update-driven
  - information from multiple heterogeneous sources is integrated in advance and stored for query
  → more efficient than heterogeneous DB
  but data (stored over longer period) are not as current

DW systems vs. Operational DBMS

OLTP (on-line transaction processing)
- Major task of operational (traditional relational) DBMS
- Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- OLTP Operations:
  - Are structured, repetitive, short, atomic and isolated transactions
  - Require detailed and up-to-date data

OLAP (on-line analytical processing)
- Major task of data warehouse system
- Serve knowledge workers: Data analysis and decision making
- OLAP Operations:
  - View data flexibly from different perspectives and abstractions
  - Require historical data with time elements
DW systems vs. Operational DBMS

Distinct features

- User/system orientation: customer vs. market
- Data contents: current, detailed vs. historical, consolidated
- Design: ER & app-oriented vs. star & subj-oriented
- View: current, local vs. evolutionary, integrated
- Access patterns: update vs. read-only but complex queries

OLTP vs. OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>clerk, IT professional</td>
<td>knowledge worker</td>
</tr>
<tr>
<td>function</td>
<td>day to day operations</td>
<td>decision support</td>
</tr>
<tr>
<td>DB design</td>
<td>application-oriented</td>
<td>subject-oriented</td>
</tr>
<tr>
<td>data</td>
<td>current, up-to-date</td>
<td>historical, multidimensional</td>
</tr>
<tr>
<td></td>
<td>detailed, flat relational</td>
<td>summarized, multidimensional</td>
</tr>
<tr>
<td></td>
<td>isolated</td>
<td>integrated, consolidated</td>
</tr>
<tr>
<td>usage</td>
<td>repetitive</td>
<td>ad-hoc</td>
</tr>
<tr>
<td>access</td>
<td>read/write</td>
<td>lots of scans</td>
</tr>
<tr>
<td></td>
<td>indexed on prim. key</td>
<td></td>
</tr>
<tr>
<td>unit of work</td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td># records accessed</td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td>#users</td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td>DB size</td>
<td>GBM-GBB</td>
<td>TB</td>
</tr>
<tr>
<td>metric</td>
<td>transaction throughput</td>
<td>query throughput, response</td>
</tr>
</tbody>
</table>

Why Separate Data Warehouse?

- High performance is desired in both DBMS and DW systems
  - DBMS – tuned for OLTP: indexing, searching certain records, querying “canned” queries
  - DW – tuned for OLAP: complex OLAP queries, use of special organization, multidimensional view, consolidation.
  - OLAP necessitates special data organization that supports multidimensional views
    → If mixed - OLAP queries would degrade operational DBMS
- Different functions, contents and uses in both systems
  - DBMS – Transactional operations, requires concurrency controls (to ensure data consistency)
    → OLAP is read only and does not need concurrency control.
    Concurrency control on OLAP’s operations would jeopardize execution of concurrent transactions in DBMS
  - DW – Decision support, requires consolidation of data from heterogeneous sources (for aggregation etc.) and historical data
    → DBMS does not maintain historical data nor facilitates data consolidation

Outline

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse design & architecture
- Data warehouse implementation
- From data warehousing to data mining
Data Cubes

- A data warehouse is based on a multi-dimensional data model which views data in the form of a data cube.
- Example: A data cube with three dimensions: item (type), time (quarter), location (city) with measure: Dollars_sold.

<table>
<thead>
<tr>
<th>Item (type)</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>oven</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dollars_sold (in thousands) of (phone, Q1, Vancouver) = 400</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dollars_sold of (TV, Q1) = 650</td>
</tr>
<tr>
<td>Dollars_sold of (Q3)</td>
</tr>
</tbody>
</table>

Lattice of Cuboids

- The lattice of cuboids forms a data cube.
- Each cuboid represents a different degree of summarization.
- The bottom most n-D base cube is called a base cuboid.
- The top most 0-D cuboid is called the apex cuboid - holds the highest-level of summarization - denoted by all.

Three data model schemas

- Star schema: A fact table in the middle connected to a set of dimension tables.
- Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into additional smaller dimension tables, forming a shape similar to snowflake.
- Fact constellation schema: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation.
Example of Star Schema

- **Time**: dimension table
  - time_key
day
day_of_the_week
month
quarter
year

- **Sales**: Fact Table
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
dollars_sold
  - avg_sales

- **Item**: dimension table
  - item_key
  - item_name
  - brand
type
  - supplier_type

- **Branch**: dimension table
  - branch_key
  - branch_name
  - branch_type

- **Location**: dimension table
  - location_key
  - street
city
  - state_or_province
country

Star → Snowflake

- **Star introduces redundancy**
  E.g., (203 Pine St, Abilene, TX, USA)
  (205 14th St, Abilene, TX, USA)

- **Snowflake normalizes this dimension table to**
  (203 Pine St, Abilene_key)
  (205 14th St, Abilene_key)
  (Abilene_key, TX, USA)

Example of Snowflake Schema

- **Time**: dimension table
  - time_key
day
day_of_the_week
month
quarter
year

- **Sales**: Fact Table
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
dollars_sold
  - avg_sales

- **Item**: dimension table
  - item_key
  - item_name
  - brand
type
  - supplier_type

- **Branch**: dimension table
  - branch_key
  - branch_name
  - branch_type

- **Location**: dimension table
  - location_key
  - street
city
  - state_or_province
country

Example of Constellation Schema

- **Time**: dimension table
  - time_key
day
day_of_the_week
month
quarter
year

- **Sales**: Fact Table
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
dollars_sold
  - avg_sales

- **Item**: dimension table
  - item_key
  - item_name
  - brand
type
  - supplier_type

- **Branch**: dimension table
  - branch_key
  - branch_name
  - branch_type

- **Location**: dimension table
  - location_key
  - street
city
  - state_or_province
country

- **Shipping**: Fact Table
  - time_key
  - shipper_key
  - from_location
to_location
dollars_cost
  - units_shipped

- **Shipper**: dimension table
  - shipper_key
  - shipper_name
  - location_key
  - shipper_type
Which design schema to use?

- Benchmarking performance can be used to select the best schema
- Snowflake vs. Star
  - Snowflake: easier to maintain dimension tables when they are very large (reduce space)
  - Star: more effective for browsing (less joins for query)

Data Warehouse: uses “fact constellation” since it can model multiple, interrelated subjects
Data Mart: uses “star” (more popular) or “snowflake”

DMQL: A Data Mining Query Language

- Cube Definition (Fact Table)
  ```dmql```
  define cube <cube_name> [dimension_list]: measure_list
  ```
- Dimension Definition (Dimension Table)
  ```dmql```
  define dimension <dimension_name> as (attribute_or_subdimension_list)
  ```
- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  ```dmql```
  define dimension <dimension_name> as <dimension_name_first_time> in cube
  ```
- Example of Star Schema

<table>
<thead>
<tr>
<th>Time: dimension table</th>
<th>Item: dimension table</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_key</td>
<td>item_key</td>
</tr>
<tr>
<td>day</td>
<td>item_name</td>
</tr>
<tr>
<td>day_of_the_week</td>
<td>brand</td>
</tr>
<tr>
<td>month</td>
<td>supplier_type</td>
</tr>
<tr>
<td>quarter</td>
<td></td>
</tr>
<tr>
<td>year</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sales: Fact Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_key</td>
</tr>
<tr>
<td>item_key</td>
</tr>
<tr>
<td>branch_key</td>
</tr>
<tr>
<td>location_key</td>
</tr>
<tr>
<td>units_sold</td>
</tr>
<tr>
<td>dollars_sold</td>
</tr>
<tr>
<td>avg_sales</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Branch: dimension table</th>
</tr>
</thead>
<tbody>
<tr>
<td>branch_key</td>
</tr>
<tr>
<td>branch_name</td>
</tr>
<tr>
<td>branch_type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>location_key</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state_or_province</td>
</tr>
<tr>
<td>country</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>location_key</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>province_or_state</td>
</tr>
<tr>
<td>country</td>
</tr>
</tbody>
</table>

A Star Schema in DMQL

```dmql```
define cube sales [time, item, branch, location]:
  dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
  units_sold = count(*)
```
**Example of Snowflake Schema**

- **Time**: dimension table
  - `time_key`
  - `day`
  - `day_of_the_week`
  - `month`
  - `quarter`
  - `year`

- **Location**: dimension table
  - `location_key`
  - `street`
  - `City_key`

- **Branch**: dimension table
  - `branch_key`
  - `branch_name`
  - `branch_type`

- **Sales**: Fact Table
  - `time_key`
  - `item_key`
  - `branch_key`
  - `location_key`
  - `units_sold`
  - `dollars_sold`
  - `avg_sales`

- **Item**: dimension table
  - `item_key`
  - `item_name`
  - `brand`
  - `type`
  - `supplier_key`

- **Measures**: `dollars_sold`, `avg_sales`

**A Snowflake Schema in DMQL**

```
define cube sales [time, item, branch, location]:
  dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
  units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier (supplier-key, supplier_type))
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city (city-key, province_or_state, country))
```

**Example of Constellation Schema**

- **Time**: dimension table
  - `time_key`
  - `day`
  - `day_of_the_week`
  - `month`
  - `quarter`
  - `year`

- **Location**: dimension table
  - `location_key`
  - `street`
  - `city`
  - `state_or_province`
  - `country`

- **Branch**: dimension table
  - `branch_key`
  - `branch_name`
  - `branch_type`

- **Sales**: Fact Table
  - `time_key`
  - `item_key`
  - `branch_key`
  - `location_key`
  - `units_sold`
  - `dollars_sold`
  - `avg_sales`

- **Item**: dimension table
  - `item_key`
  - `item_name`
  - `brand`
  - `type`
  - `supplier_type`

- **Measures**: `dollars_sold`, `avg_sales`

- **Shipping**: Fact Table
  - `shipper_key`
  - `from_location`
  - `to_location`
  - `dollars_cost`
  - `units_shipped`

- **Shipper**: dimension table
  - `shipper_key`
  - `shipper_name`
  - `location_key`

- **Location**: dimension table
  - `location_key`
  - `street`
  - `city`
  - `state_or_province`
  - `country`

**A fact constellation Schema in DMQL**

```
define cube sales [time, item, branch, location]:
  dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
  units_sold = count(*)

define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier (supplier-key, supplier_type))
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city (city-key, province_or_state, country))
define cube shipping [time, item, shipper, from_location, to_location]:
  dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)

define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension branch as branch in cube sales
define dimension location as location in cube sales
define dimension shipper as shipper in cube sales
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
```
Measures

Quantities computed by aggregating values in data cube cells in various ways:

- **distributive**: if the result can be derived by aggregating results from partitioned data.
  - E.g., count(), sum(), min(), max().
- **algebraic**: if it can be computed by an algebraic function with a finite number of arguments, each of which is obtained by applying a distributive function.
  - E.g., avg() = sum()/count(), standard_deviation().
- **holistic**: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank().

Concept Hierarchy

Example: Hierarchy for the dimension “location”

- **all**
  - **region**
    - Europe
    - North America
  - **country**
    - Germany
    - Spain
    - Canada
    - Mexico
  - **city**
    - Frankfurt
    - Vancouver
    - Toronto
  - **office**
    - L. Chan
    - M. Wind

Specifications of Hierarchies

- **Schema hierarchy**:
  - Office < City < Country < Region
  - Day < (month<quarter; week) < year

- **Set_grouping hierarchy**
  - (1..10] < inexpensive

Hierarchy and Warehouse View

DBMiner:
- Importing data
- Table browsing
- Dimension creation/browsing
- Cube building/browsing
Building cube:
Sales volume as a function of product, month, and region

Time (quarter)
1Qtr 2Qtr 3Qtr 4Qtr

Location (city)
U.S.A  Canada  Mexico

Product (type)
TV  PC  VCR  sum

sum

Total annual sales of TVs in USA

ALL = Total sales

Browsing a Data Cube
- Visualization
- OLAP capabilities
- Interactive manipulation

Typical OLAP operations
- **Roll up** (increase the level of abstraction):
  - Summarize data by climbing up hierarchy or by dimension reduction
- **Drill down** (decrease the level of abstraction):
  - Reverse of roll-up from higher level summary to lower level summary or detailed data, or introducing new dimensions

Examples: Roll-up & drill-down
Typical OLAP operations (cont)

- **Slice and dice** (project and select):
- **Pivot** (rotate):
  - reorient the cube, visualization, 3D to series of 2D planes.
- Other operations
  - **drill across** (links to the raw data): involving (across) more than one fact table
  - **drill through**: through the bottom level of the cube to its back-end relational tables (using SQL)

OLAP functions

- OLAP offers functions to:
  - Calculate e.g., ratios, variance, etc.
  - Generate e.g., summarizations, aggregations and hierarchies at each level and every dimension intersection
  - Support forecasting functions e.g., trend and stat analysis
- OLAPs and SDBs (statistical Databases) are similar
  - SDBs focus on socioeconomic applications, and are sensitive to privacy issues when drilling down
  - OLAPs are designed to handle HUGE amounts of data
- OLAP’s querying is based on a **starnet model** – representing abstraction levels in each dimension

Examples: dice, slice, pivot

A Star-Net Query Model

Each circle is called a **footprint**
Outline

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse design & architecture
- Data warehouse implementation
- From data warehousing to data mining

A Data Warehouse Design

Four views to be considered:
- **Top-down view**: allows selection of the relevant information necessary for the data warehouse
- **Data source view**: exposes the information being captured, stored, and managed by operational systems
- **Data warehouse view**: consists of fact tables and dimension tables
- **Business query view**: perspectives of data in the warehouse from the view of end-users

Design approaches

- **Top-down**: Starts with overall design and planning
  - good for mature technology and well-defined problems
  - Systematic and minimal integration problem
  - expensive and lack flexibility
- **Bottom-up**: Starts with experiments and prototypes
  - Rapid returns, low cost
  - Flexible but hard to integrate
- **Combined**

Design process

Typical steps in data warehouse design process:
- Select
  - **business process** to model
    (e.g., orders, invoices, etc.)
  - **grain** (**atomic level of data**) of the business process
    (e.g., individual transactions, daily-snapshots etc.)
  - **dimensions** that will apply to each fact table record
  - **measure** that will populate each fact table record
Three data warehouse models

- **Enterprise warehouse**
  - Collects all of the information about subjects spanning the entire organization.

- **Data Mart**
  - A subset of corporate-wide data that is of value to a specific group of users. Its scope is confined to specific, selected groups, such as marketing data mart.

- **Virtual warehouse**
  - A set of views over operational databases.
  - Only some of the possible summary views may be materialized.

Data warehouse development

As in Software Engineering, two development methods:

- **Waterfall**: do structured and systematic analysis at each step before proceeding to the next.

- **Spiral**: rapid generation of increasingly functional systems → short turn around time.
  - **Recommended** for the development of data warehouses and data marts.

A recommended approach

For data warehouse development

- Define a high-level corporate data model.

Data Warehouse Architecture

- **Data Sources**
- **Data Warehouse Server**
- **OLAP Engine**
- **Front-End (Client) Tools**

- **Bottom tier**
- **Middle tier**
- **Top tier**

- **Analysis**
- **Query Reports**
- **Data mining**
Data sources

Data sources are
- Often the operational DBMS
- Designed for operational use, not for decision support

Multiple data sources
- Are often from different systems using different hardware and customized software
- Create problems e.g., semantic conflicts

Back-End Tools & Utilities

- Data Cleaning: eliminates noise and errors. Three classes of tools:
  - Migration: simple data transformation
  - Scrubbing: by using domain-specific knowledge
  - Auditing: detects outliers
- Data Extraction: by a program interface called gateway
- Data Transformation: convert data from legacy or host format to warehouse format
- Load: sort, summarize, consolidate, compute views, check integrity constraints, and build indices and partitions
- Refresh: propagates updates from data sources to the data store
  - When to refresh? Depends on usage and types of data
  - How to refresh? Data shipping (by triggers) and Transaction shipping (by updating transaction logs)

The Bottom tier: warehouse server

- Monitor:
  - Detect changes that are of interest
  - Propagate the change to the integrator
- Integrator:
  - Integrate changes into the warehouse
  - Make the data conform to conceptual schema in the warehouse
  - Resolve possible update anomalies
- Metadata Repository: data about data (warehouse objects)
  - Administrative metadata: source DBs and their contents, security
  - Business metadata: business terms, ownership of data
  - Operational metadata: history of migrated data and transformations applied

The Middle tier: OLAP server

Implemented by:

ROLAP (Relational OLAP) e.g., microstrategy’s DSS Server, Informix’s Metacube
- Uses relational or extended-relational DBMS (often in a star schema)
- Includes optimization of DBMS backend tools
- Supports extended SQL (Sequence Query Language)
- A cell in a multi-dimensional structure is represented by a tuple
- Advantage: scalable (no empty cell for sparse cube)
- Disadvantage: no direct access to cells
The Middle tier: OLAP server (cont)

MOLAP (Multidimensional OLAP) e.g., Arbor’s Essbase
- Stores data in a multi-dimensional data structure (MDDS)
- Uses Array-based multidimensional storage engine (sparse matrix techniques)
  - **Advantage:** fast indexing to pre-computed aggregations.
  - **Disadvantage:** not very scalable and sparse

HOLAP (Hybrid OLAP)
- Scalability of ROLAP + efficiency of MOLAP
- User flexibility, e.g., low level: relational, high-level: array-based

Specialized SQL servers
- specialized support for SQL queries over star/snowflake schemas

Outline
- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse design & architecture
- Data warehouse implementation
- From data warehousing to data mining

Efficient Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains an empty cell
  - Suppose that at most one level of each dimension appears in a cuboid.

  How many cuboids in an n-dimensional cube with L levels?
  \[ T = \sum_{i=1}^{n} (L_i + 1) \]

  Ex: A data cube of 3 dimensions: time, location, item
  The total number of cuboids = 2x2x2
  The cuboid contains time or none

  Suppose now time has 3 levels of hierarchy: month < quarter < year
  The total number of cuboids = 4x2x2
  The cuboid contains month or quarter or year or none

Partial Materialization

- Pre-computing and materializing all possible cuboids of a data cube is not feasible when there are large number of them
- Given a base cuboid, there are three choices
  - **Full materialization** – pre-compute every cuboid
  - **No materialization** – do not pre-compute any “nonbase” cuboid
  - **Partial materialization** – select subset of cuboids to compute
    - Selection of which cuboids to materialize depends on the queries in the workload, their frequencies, and access costs etc.
    - E.g., OLAP product uses heuristic – “Prefer cuboids that are most referred”
Full materialization

- To ensure fast on-line processing, full materialization may be necessary → Efficient cube computation
- ROLAP cube computation
  - Optimization ideas:
    - Reorder or cluster related tuples
    - Partial grouping of subaggregates and use them to speedup computation of other subaggregates and aggregates
- MOLAP cube computation
  → Multi-way array aggregation
  (can’t use ROLAP’s approach since it is based on different data structure and data access/index from ROLAP)

Multi-way Array Aggregation

- Partition arrays into chunks (a small subcube which fits in memory).
- Compress chunks to remove wasted space -- handle sparse cubes
- Compute aggregates in “multiway” by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost.

What is the best traversing order to do multi-way aggregation?
Multi-way Array Aggregation

- **Method:** the planes should be sorted and computed from size small to large (see Han and Kamber, Example 2.12 pp. 75-78)
- **Idea:** keep the smallest plane in the main memory, fetch and compute only one chunk at a time for the largest plane
- **Limitation:** efficient for a small number of dimensions
  - If there is a large number of dimensions,
    - bottom-up computation [Beyer & Ramakrishnan, SIGMOD’99]
    - iceberg cube computation

Indexing OLAP data

**Bitmap Indexing**
- Index on a particular column (attribute)
- Each value in the column corresponds to a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The $i$-th bit is set if the $i$-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains
Indexing OLAP data

Join index
- Allows records to identify joinable tuples without performing join operation (which is expensive) - used to identify subcubes of interest
  E.g., in star schema data warehouse
  Fact table, “sale” and dimensions: location, item

Example: Selecting cuboids

Which cuboid to process query on {brand, state} with “year = 2000”?
& cuboid1: {item_name, city, year}
  cuboid2: {brand, country, year}
  cuboid3: {brand, state, year}
  cuboid4: {item_name, state} where year = 2000
- Eliminate cuboid2
  since “country” can’t be generalized to “state” (lower level)
- By generalization rule, the rest of cuboids will work
  But which is better?
  cuboid1: requires two generalizations → cost most
  if not many year values associated with brand but each brand has several
  item_names → couboid3 is less costly than cuboid4

Outline
- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse design & architecture
- Data warehouse implementation
- From data warehousing to data mining

Data Warehouse Usage
- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools.
- Differences among the three tasks
From OLAP to OLAM

- Why OLAM (online analytical mining)?
  - High quality of data in data warehouses (DW)
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
  - OLAP gives a good framework for data exploration
    - "interactive" mining with drilling, dicing, pivoting, etc.
    - On-line selection of data mining functions e.g., integration and swapping of multiple mining functions, algorithms, and tasks.

Discovery-Driven Exploration

Data cube exploration:
- Hypothesis-driven
  - exploration by user, huge search space, can overlook
- Discovery-driven (Sarawagi, et al.’98)
  - Effective navigation of large OLAP data cubes
  - pre-compute measures indicating exceptions, guide user in the data analysis, at all levels of aggregation
- Exception: significantly different from the value anticipated, based on a statistical model
- Visual cues such as background color are used to reflect the degree of exception of each cell

OLAM Architecture

Example
“Exception” and its computation

- Parameters
  - SelfExp: surprise of cell relative to other cells at same level of aggregation
  - InExp: surprise beneath the cell
  - PathExp: surprise beneath cell for each drill-down path
- Computation of exception indicator (model fitting and computing SelfExp, InExp, and PathExp values) can be overlapped with cube construction
- Exception themselves can be stored, indexed and retrieved like pre-computed aggregates

Data Mining & Warehouses

- Construction of data warehouses requires data cleaning and integration
  → important preprocessing in data mining
- Data mining tools can interface with the OLAP engine
  → facilitates interactive data exploration, including data integration, aggregation and navigation
- OLAP-based mining → OLAM

Summary

- Data warehouse
  - A multi-dimensional model of a data warehouse
    - Representation: A data cube consists of dimensions & measure
    - Schema: Star, snowflake, and fact constellations
  - OLAP operations: drilling, rolling, slicing, dicing and pivoting
  - OLAP servers: ROLAP, MOLAP, HOLAP
  - Efficient computation of data cubes
    - Partial vs. full vs. no materialization
    - Multi-way array aggregation
    - Bitmap index and join index implementations
  - Data mining and data warehouse
    - Discovery-driven in data warehouse
    - From OLAP to OLAM (on-line analytical mining)

For more, see …

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB’96
- G. Dong, J. Han, J. Lam, J. Pei, K. Wang. Mining Multi-dimensional Constrained Gradients in Data Cubes. VLDB’2001
For more, see …

- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD’01
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD’96
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD’97.