Chapter 3: Data Warehousing and OLAP Technology: An Overview

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

- Defined in many different ways, but not rigorously.
  - A decision support database that is maintained *separately* from the organization’s operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis.
- “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
- Data warehousing:
  - The process of constructing and using data warehouses
Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process
Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
- When data is moved to the warehouse, it is converted.
Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain “time element”
Data Warehouse—Nonvolatile

- A *physically separate store* of data transformed from the operational environment
- Operational *update of data does not occur* in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
- Requires only two operations in data accessing:
  - *initial loading of data* and *access of data*
Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration: A query driven approach
  - Build wrappers/mediators on top of heterogeneous databases
  - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
  - Complex information filtering, compete for resources
- Data warehouse: update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis
Data Warehouse vs. Operational DBMS

- **OLTP (on-line transaction processing)**
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.

- **OLAP (on-line analytical processing)**
  - Major task of data warehouse system
  - Data analysis and decision making

- **Distinct features (OLTP vs. OLAP):**
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - **Database design**: ER with application vs. Star Schema with subject
  - View: current, local vs. evolutionary, integrated
  - Access patterns: update vs. read-only but complex queries
Chapter 3: Data Warehousing and OLAP Technology: An Overview

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture
- Data warehouse implementation
- From data warehousing to data mining
From Tables and Spreadsheets to Data Cubes

- A data warehouse is based on a **multidimensional data model** which views data in the form of a data cube.
- A data cube, such as **sales**, allows data to be modeled and viewed in multiple dimensions:
  - Dimension tables, such as **item (item_name, brand, type)**, or **time(day, week, month, quarter, year)**
  - Fact table contains measures (such as **dollars_sold**) and keys to each of the related dimension tables.
- In data warehousing literature, an n-D base cube is called a **base cuboid**. The top most 0-D cuboid, which holds the highest-level of summarization, is called the **apex cuboid**. The lattice of cuboids forms a **data cube**.
Cube: A Lattice of Cuboids

0-D (apex) cuboid
1-D cuboids
2-D cuboids
3-D cuboids
4-D (base) cuboid
Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  - **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 a set of smaller dimension tables, forming a shape similar to snowflake
  - **Fact constellations**: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation
Example of Star Schema

Sales Fact Table

time_key
item_key
branch_key
location_key
units_sold
dollars_sold
avg_sales

time
time_key
day
day_of_the_week
month
quarter
year

item
item_key
item_name
brand
type
supplier_type

branch
branch_key
branch_name
branch_type

location
location_key
street
city
state_or_province
country
Example of Snowflake Schema
Example of Fact Constellation

Sales Fact Table

- time_key
- item_key
- branch_key
- location_key
- units_sold
- dollars_sold
- avg_sales

Measures

Shipping Fact Table

- time_key
- item_key
- shipper_key
- from_location
- to_location
- dollars_cost
- units_shipped

item

- item_key
- item_name
- brand
- type
- supplier_type

location

- location_key
- street
- city
- province_or_state
- country

shipper

- shipper_key
- shipper_name
- location_key
- shipper_type
Cube Definition Syntax (BNF) in DMQL

- Cube Definition (Fact Table)
  ```
  define cube <cube_name> [<dimension_list>]:
  <measure_list>
  ```

- Dimension Definition (Dimension Table)
  ```
  define dimension <dimension_name> as
  (<attribute_or_subdimension_list>)
  ```

- Special Case (Shared Dimension Tables)
  - First time as “cube definition”
  ```
  define dimension <dimension_name> as
  <dimension_name_first_time> in cube
  <cube_name_first_time>
  ```
Defining Star Schema in DMQL

define cube sales_star [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_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city,
    province_or_state, country)
Defining Snowflake Schema in DMQL

define cube sales_snowflake [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))
Defining Fact Constellation 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_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, 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 shipper as (shipper_key, shipper_name, location as location
  in cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales
Measures of Data Cube: Three Categories

- **Distributive**: if the result derived by applying the function to \( n \) aggregate values (\( n \) partitions) is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()

- **Algebraic**: if it can be computed by an algebraic function with \( M \) arguments (where \( M \) is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg() = sum() / count(), min_N(), standard_deviation()

- **Holistic**: if there is no constant bound on the storage size (ex: \( M \) tuples) needed to describe a subaggregate.
  - E.g., median(), mode(), rank(), MostFrequent()
A Concept Hierarchy: Dimension (location)

- all
  - region
    - country
      - city
        - office
  - Europe
    - Germany
      - Frankfurt
    - Spain
    - Canada
      - Vancouver
    - North_America
      - Toronto
      - Mexico
    - ...
View of Warehouses and Hierarchies

Specification of hierarchies

- Schema hierarchy
  
  `day < {month < quarter; week} < year`

- Set_grouping hierarchy
  
  `{1..10} < inexpensive`
Multidimensional Data

- Sales volume as a function of product, month, and region

Dimensions: Product, Location, Time
Hierarchical summarization paths

Industry  Region  Year
Category  Country  Quarter
Product  City  Month  Week
Office  Day
A Sample Data Cube

- Total annual sales of TV in U.S.A.
Cuboids Corresponding to the Cube

0-D (apex) cuboid
1-D cuboids
2-D cuboids
3-D (base) cuboid
Browsing a Data Cube

- Visualization
- OLAP capabilities
- Interactive manipulation
Fig. 3.10 Typical OLAP Operations
Typical OLAP Operations

- Roll up (drill-up): summarize data
  - by climbing up hierarchy or by dimension reduction
- Drill down (roll down): reverse of roll-up
  - from higher level summary to lower level summary or detailed data, or introducing new dimensions
- Slice and dice: project and select
- Pivot (rotate):
  - reorient the cube, visualization, 3D to series of 2D planes
- Other operations
  - drill across: involving (across) more than one fact table
  - drill through: through the bottom level of the cube to its back-end relational tables (using SQL)
A Star-Net Query Model

Each circle is called a footprint
Chapter 3: Data Warehousing and OLAP Technology: An Overview

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture
- Data warehouse implementation
- From data warehousing to data mining
Design of Data Warehouse: A Business Analysis Framework

- Four views regarding the design of a data warehouse
  - **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**
    - sees the perspectives of data in the warehouse from the view of end-user
Data Warehouse Design Process

- Top-down, bottom-up approaches or a combination of both
  - **Top-down**: Starts with overall design and planning (mature)
  - **Bottom-up**: Starts with experiments and prototypes (rapid)
- From software engineering point of view
  - **Waterfall**: structured and systematic analysis at each step before proceeding to the next
  - **Spiral**: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
  - Choose a **business process** to model, e.g., orders, invoices, etc.
  - Choose the **grain (atomic level of data)** of the business process
  - Choose the **dimensions** that will apply to each fact table record
  - Choose the **measure** that will populate each fact table record
Data Warehouse: A Multi-Tiered Architecture

Data Sources
- Operational DBs
- Other sources

Data Storage
- Data Warehouse
- Data Marts

OLAP Engine
- Monitor & Integrator
- OLAP Server

Front-End Tools
- Analysis
- Query
- Reports
- Data mining

Other sources
- Metadata

Operational DBs
- Extract
- Transform
- Load
- Refresh

Data Marts
- Serve
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 groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
    - Independent vs. dependent (directly from warehouse) data mart

- **Virtual warehouse**
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized
Data Warehouse Development: A Recommended Approach

Define a high-level corporate data model

Distributed Data Marts

Data Mart

Data Mart

Model refinement

Model refinement

Multi-Tier Data Warehouse

Enterprise Data Warehouse
Data Warehouse Back-End Tools and Utilities

- Data extraction
  - get data from multiple, heterogeneous, and external sources
- Data cleaning
  - detect errors in the data and rectify them when possible
- Data transformation
  - convert data from legacy or host format to warehouse format
- Load
  - sort, summarize, consolidate, compute views, check integrity, and build indices and partitions
- Refresh
  - propagate the updates from the data sources to the warehouse
Metadata Repository

- Meta data is the data defining warehouse objects. It stores:
- Description of the structure of the data warehouse
  - schema, view, dimensions, hierarchies, derived data defn, data mart locations and contents
- Operational meta-data
  - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarization
- The mapping from operational environment to the data warehouse
- Data related to system performance
  - warehouse schema, view and derived data definitions
- Business data
  - business terms and definitions, ownership of data, charging policies
OLAP Server Architectures

- **Relational OLAP (ROLAP)**
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability

- **Multidimensional OLAP (MOLAP)**
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data

- **Hybrid OLAP (HOLAP)** (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array

- **Specialized SQL servers** (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas
Chapter 3: Data Warehousing and OLAP Technology: An Overview

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse architecture
- Data warehouse implementation
- From data warehousing to data mining
Fig. 3.10 Typical OLAP Operations
Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*
- Slice and dice: *project and select*
- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes*
- Other operations
  - *drill across: involving (across) more than one fact table*
  - *drill through: through the bottom level of the cube to its back-end relational tables (using SQL)*
Efficient Data 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 only one cell
  - How many cuboids in an n-dimensional cube with L levels?
    \[ T = \prod_{i=1}^{n} (L_i + 1) \]

- Materialization of data cube
  - Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)

- Selection of which cuboids to materialize
  - Based on size, sharing, access frequency, etc.
Cube Operation

- Cube definition and computation in DMQL
  
  **define cube** sales[item, city, year]: sum(sales_in_dollars)

  **compute cube** sales

- Transform it into a SQL-like language (with a new operator **cube by**, introduced by Gray et al.’96)

  SELECT item, city, year, SUM(amount)
  FROM SALES

  **CUBE BY** item, city, year

- Need compute $2^3$ Group-Bys for Cube By on 3 dim -- date, product, customer

  ( For n dim, $2^n$ Group-Bys)

  (date, product, customer), $\rightarrow$ base cube (already computed) from which all the other cubes computed

  (date, product), (date, customer), (product, customer), (date), (product), (customer)

  () $\rightarrow$ 0-D apex cuboid : all all all

- N attributes to aggregate in Select, $2^N - 1$ of All's

- $C_i$ is cardinality of i attribute, then cardinality of cube = $(C_1 + 1) \times (C_2 + 1) \times \cdots \times (C_N + 1)$ where + 1 is for All
Summarizing Data Using CUBE, ROLLUP

SQL Server 2005

- The ROLLUP operator is useful in generating reports that contain subtotals and totals. The ROLLUP operator generates a result set that is similar to the result sets generated by the CUBE operator.
- Following are the specific differences between CUBE and ROLLUP:
  - CUBE generates a result set that shows aggregates for all combinations of values in the selected columns.
  - ROLLUP generates a result set that shows aggregates for a hierarchy of values in the selected columns.
- For example, a simple table *Inventory* contains the following:

<table>
<thead>
<tr>
<th>Item</th>
<th>Color</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Blue</td>
<td>124</td>
</tr>
<tr>
<td>Table</td>
<td>Red</td>
<td>223</td>
</tr>
<tr>
<td>Chair</td>
<td>Blue</td>
<td>101</td>
</tr>
<tr>
<td>Chair</td>
<td>Red</td>
<td>210</td>
</tr>
</tbody>
</table>
Summarizing Data Using CUBE

This query generates a subtotal report:

```sql
SELECT CASE WHEN (GROUPING( Item ) = 1) THEN 'ALL'
    ELSE ISNULL(Item, 'UNKNOWN')
END AS Item,
CASE WHEN (GROUPING( Color ) = 1) THEN 'ALL'
    ELSE ISNULL(Color, 'UNKNOWN')
END AS Color,
SUM(Quantity) AS QtySum
FROM Inventory
GROUP BY Item, Color WITH CUBE;
```
### Summarizing Data Using CUBE

<table>
<thead>
<tr>
<th>Item</th>
<th>Color</th>
<th>QtySum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>Blue</td>
<td>101.00</td>
</tr>
<tr>
<td>Chair</td>
<td>Red</td>
<td>210.00</td>
</tr>
<tr>
<td>Chair</td>
<td>ALL</td>
<td>311.00</td>
</tr>
<tr>
<td>Table</td>
<td>Blue</td>
<td>124.00</td>
</tr>
<tr>
<td>Table</td>
<td>Red</td>
<td>223.00</td>
</tr>
<tr>
<td>Table</td>
<td>ALL</td>
<td>347.00</td>
</tr>
<tr>
<td>ALL</td>
<td>Blue</td>
<td>225.00</td>
</tr>
<tr>
<td>ALL</td>
<td>Red</td>
<td>433.00</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>658.00</td>
</tr>
</tbody>
</table>
Summarizing Data Using ROLL UP

- This query generates a subtotal report:

```sql
SELECT CASE WHEN (GROUPING(Item) = 1) THEN 'ALL'
             ELSE ISNULL(Item, 'UNKNOWN')
END AS Item,
CASE WHEN (GROUPING(Color) = 1) THEN 'ALL'
             ELSE ISNULL(Color, 'UNKNOWN')
END AS Color,
SUM(Quantity) AS QtySum
FROM Inventory
GROUP BY Item, Color WITH ROLL UP
```
### Summarizing Data Using Roll Up

<table>
<thead>
<tr>
<th>Item</th>
<th>Color</th>
<th>QtySum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>Blue</td>
<td>101.00</td>
</tr>
<tr>
<td>Chair</td>
<td>Red</td>
<td>210.00</td>
</tr>
<tr>
<td>Chair</td>
<td>ALL</td>
<td>311.00</td>
</tr>
<tr>
<td>Table</td>
<td>Blue</td>
<td>124.00</td>
</tr>
<tr>
<td>Table</td>
<td>Red</td>
<td>223.00</td>
</tr>
<tr>
<td>Table</td>
<td>ALL</td>
<td>347.00</td>
</tr>
<tr>
<td>ALL</td>
<td>ALL</td>
<td>658.00</td>
</tr>
</tbody>
</table>
Analytic Functions

- Analytic functions compute an aggregate value based on a group of rows.
- Last Operation to Apply: All joins and all WHERE, GROUP BY, and HAVING clauses are completed before the analytic functions are processed; Only appear in Select, Order By Clause.
- They differ from aggregate functions in that they return multiple rows for each group.
- The group of rows is called a window and is defined by the `analytic_clause`:
  - For each row, a sliding window of rows is defined.
  - The window determines the range of rows used to perform the calculations for the current row.
- `PARTITION BY` { value_expr[, value_expr ]... | ( value_expr[, value_expr ]... )}
- `ORDER BY` to specify how data is ordered within a partition.
Analytic Functions: Window aggregate functions

Function(arg) OVER (partition-clause, order-clause, window-agg-group);

- The OVER clause specifies the three primary attributes of the function. These three attributes are optional.
- The order-clause is like an ORDER BY clause of a statement except that the order is only relevant in the context of the function.
- The partition-clause is similar to the commonly used GROUP BY clause but again is relevant only in the context of the function.
- The window-agg-group clause allows the specification of a window of rows to which the aggregation is applied.
Analytic Functions: Window aggregate functions

```
SELECT empnum, dept, salary,
    SUM(salary) OVER (partition by dept) AS deptsum,
    DECIMAL(salary,17,0) * 100 / SUM(salary) OVER (partition by dept) AS salratio
FROM employee;
```

<table>
<thead>
<tr>
<th>EMPNUM</th>
<th>DEPT</th>
<th>SALARY</th>
<th>DEPTSUM</th>
<th>SALRATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>78000</td>
<td>383000</td>
<td>20.365</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>75000</td>
<td>383000</td>
<td>19.582</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>75000</td>
<td>383000</td>
<td>19.582</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>53000</td>
<td>383000</td>
<td>13.838</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>52000</td>
<td>383000</td>
<td>13.577</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>50000</td>
<td>383000</td>
<td>13.054</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-</td>
<td>51000</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>51000</td>
<td>51000</td>
<td>100.000</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>79000</td>
<td>209000</td>
<td>37.799</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>75000</td>
<td>209000</td>
<td>35.885</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>55000</td>
<td>209000</td>
<td>26.315</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
<td>84000</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>84000</td>
<td>84000</td>
<td>100.000</td>
</tr>
</tbody>
</table>
Analytic Functions: Window aggregate functions

```
SELECT empno, deptno, hiredate,
       ROW_NUMBER( ) OVER (PARTITION BY deptno ORDER BY hiredate NULLS LAST) SRLNO
FROM emp
WHERE deptno IN (10, 20)
ORDER BY deptno, SRLNO;
```

`ROW_NUMBER( )` gives a running serial number to a partition of records.

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>DEPTNO</th>
<th>HIREDATE</th>
<th>SRLNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>7782</td>
<td>10</td>
<td>09-JUN-81</td>
<td>1</td>
</tr>
<tr>
<td>7839</td>
<td>10</td>
<td>17-NOV-81</td>
<td>2</td>
</tr>
<tr>
<td>7934</td>
<td>10</td>
<td>23-JAN-82</td>
<td>3</td>
</tr>
<tr>
<td>7369</td>
<td>20</td>
<td>17-DEC-80</td>
<td>1</td>
</tr>
<tr>
<td>7566</td>
<td>20</td>
<td>02-APR-81</td>
<td>2</td>
</tr>
<tr>
<td>7902</td>
<td>20</td>
<td>03-DEC-81</td>
<td>3</td>
</tr>
<tr>
<td>7788</td>
<td>20</td>
<td>09-DEC-82</td>
<td>4</td>
</tr>
<tr>
<td>7876</td>
<td>20</td>
<td>12-JAN-83</td>
<td>5</td>
</tr>
</tbody>
</table>
Iceberg Cube

- Computing only the cuboid cells whose count or other aggregates satisfying the condition like

  \[
  \text{HAVING COUNT(*) } \geq \text{ minsup}
  \]

- Motivation
  - Only a small portion of cube cells may be “above the water” in a sparse cube
  - Only calculate “interesting” cells—data above certain threshold
  - Avoid explosive growth of the cube
    - Suppose 100 dimensions, only 1 base cell. How many aggregate cells if count \( \geq 1 \)? What about count \( \geq 2 \)?
Indexing OLAP Data: **Bitmap Index**

- Index on a particular column
- Each value in the column has 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
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS’06]

<table>
<thead>
<tr>
<th>Cust</th>
<th>Region</th>
<th>Type</th>
<th>Index on Region</th>
<th>Index on Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Asia</td>
<td>Retail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>Europe</td>
<td>Dealer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>Asia</td>
<td>Dealer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>America</td>
<td>Retail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Europe</td>
<td>Dealer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Base table

<table>
<thead>
<tr>
<th>RecID</th>
<th>Asia</th>
<th>Europe</th>
<th>America</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Index on Region

<table>
<thead>
<tr>
<th>RecID</th>
<th>Retail</th>
<th>Dealer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Indexing OLAP Data: Join Indices

- Join index: $JI(R\text{-id}, S\text{-id})$ where $R (R\text{-id}, ...)$ $\bowtie S (S\text{-id}, ...)$
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join
- In data warehouses, join index relates the values of the **dimensions** of a start schema to **rows** in the fact table.
  - E.g. fact table: *Sales* and two dimensions *city* and *product*
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions
Efficient Processing OLAP Queries

- **Determine which operations** should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection

- **Determine which materialized cuboid(s)** should be selected for OLAP op.
  - Let the query to be processed be on \{brand, province_or_state\} with the condition “year = 2004”, and there are 4 materialized cuboids available:
    1) \{year, item_name, city\}
    2) \{year, brand, country\}
    3) \{year, brand, province_or_state\}
    4) \{item_name, province_or_state\} where year = 2004
      Which should be selected to process the query?

- Explore indexing structures and compressed vs. dense array structs in MOLAP
Chapter 3: Data Warehousing and OLAP Technology: An Overview

- What is a data warehouse?
- A multi-dimensional data model
- Data warehouse 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
From On-Line Analytical Processing (OLAP) to On Line Analytical Mining (OLAM)

- Why online analytical mining?
  - High quality of data in data warehouses
    - DW contains integrated, consistent, cleaned data
  - Available information processing structure surrounding data warehouses
    - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
  - OLAP-based exploratory data analysis
    - Mining with drilling, dicing, pivoting, etc.
  - On-line selection of data mining functions
    - Integration and swapping of multiple mining functions, algorithms, and tasks
An OLAM System Architecture

Layer 4: User Interface
   Mining query → Mining result

Layer 3: OLAP/OLAM
   User GUI API

Layer 2: MDDB
   OLAM Engine <-> OLAP Engine
   Data Cube API
   MDDB
   Meta Data

Layer 1: Data Repository
   Databases
   Filtering & Integration
   Database API
   Filtering
   Data cleaning
   Data integration
   Data Warehouse
OLAP Server Architectures

- **Relational OLAP (ROLAP)**
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability

- **Multidimensional OLAP (MOLAP)**
  - Sparse array-based multidimensional storage engine
  - Fast indexing to pre-computed summarized data

- **Hybrid OLAP (HOLAP)** (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array

- **Specialized SQL servers** (e.g., Redbricks)
  - Specialized support for SQL queries over star/snowflake schemas
Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary
Attribute-Oriented Induction

- Proposed in 1989 (KDD ‘89 workshop)
- Not confined to categorical data nor particular measures
- How it is done?
  - Collect the task-relevant data (initial relation) using a relational database query
  - Perform generalization by attribute removal or attribute generalization
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts
  - Interaction with users for knowledge presentation
Attribute-Oriented Induction: An Example

Example: Describe general characteristics of graduate students in the University database

- Step 1. Fetch relevant set of data using an SQL statement, e.g.,
  ```sql
  Select * (i.e., name, gender, major, birth_place, birth_date, residence, phone#, gpa) 
  from student
  where student_status in {"Msc", "MBA", "PhD" }
  ```

- Step 2. Perform attribute-oriented induction

- Step 3. Present results in generalized relation, cross-tab, or rule forms
Class Characterization: An Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Major</th>
<th>Birth-Place</th>
<th>Birth_date</th>
<th>Residence</th>
<th>Phone #</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim Woodman</td>
<td>M</td>
<td>CS</td>
<td>Vancouver, BC, Canada</td>
<td>8-12-76</td>
<td>3511 Main St., Richmond</td>
<td>687-4598</td>
<td>3.67</td>
</tr>
<tr>
<td>Scott Lachance</td>
<td>M</td>
<td>CS</td>
<td>Montreal, Que, Canada</td>
<td>28-7-75</td>
<td>345 1st Ave., Richmond</td>
<td>253-9106</td>
<td>3.70</td>
</tr>
<tr>
<td>Laura Lee</td>
<td>F</td>
<td>Physics</td>
<td>Seattle, WA, USA</td>
<td>25-8-70</td>
<td>125 Austin Ave., Burnaby</td>
<td>420-5232</td>
<td>3.83</td>
</tr>
</tbody>
</table>

**Country Age range City Removed Excl, VG,**

<table>
<thead>
<tr>
<th>Gender</th>
<th>Major</th>
<th>Birth_region</th>
<th>Age_range</th>
<th>Residence</th>
<th>GPA</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Science</td>
<td>Canada</td>
<td>20-25</td>
<td>Richmond</td>
<td>Very-good</td>
<td>16</td>
</tr>
<tr>
<td>F</td>
<td>Science</td>
<td>Foreign</td>
<td>25-30</td>
<td>Burnaby</td>
<td>Excellent</td>
<td>22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gender</th>
<th>Canada</th>
<th>Foreign</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>16</td>
<td>14</td>
<td>30</td>
</tr>
<tr>
<td>F</td>
<td>10</td>
<td>22</td>
<td>32</td>
</tr>
<tr>
<td>Total</td>
<td>26</td>
<td>36</td>
<td>62</td>
</tr>
</tbody>
</table>
Basic Principles of Attribute-Oriented Induction

- **Data focusing**: task-relevant data, including dimensions, and the result is the *initial relation*

- **Attribute-removal**: remove attribute $A$ if there is a large set of distinct values for $A$ but (1) there is no generalization operator on $A$, or (2) $A$’s higher level concepts are expressed in terms of other attributes

- **Attribute-generalization**: If there is a large set of distinct values for $A$, and there exists a set of generalization operators on $A$, then select an operator and generalize $A$

- **Attribute-threshold control**: typical 2-8, specified/default

- **Generalized relation threshold control**: control the final relation/rule size
Attribute-Oriented Induction: Basic Algorithm

- **InitialRel**: Query processing of task-relevant data, deriving the *initial relation*.
- **PreGen**: Based on the analysis of the number of distinct values in each attribute, determine generalization plan for each attribute: removal? or how high to generalize?
- **PrimeGen**: Based on the PreGen plan, perform generalization to the right level to derive a “prime generalized relation”, accumulating the counts.
- **Presentation**: User interaction: (1) adjust levels by drilling, (2) pivoting, (3) mapping into rules, cross tabs, visualization presentations.
Presentation of Generalized Results

- **Generalized relation:**
  - Relations where some or all attributes are generalized, with counts or other aggregation values accumulated.

- **Cross tabulation:**
  - Mapping results into cross tabulation form (similar to contingency tables).
  - **Visualization techniques:**
    - Pie charts, bar charts, curves, cubes, and other visual forms.

- **Quantitative characteristic rules:**
  - Mapping generalized result into characteristic rules with quantitative information associated with it, e.g.,

\[
\text{grad}(x) \land \text{male}(x) \Rightarrow \text{birth}_\text{region}(x) = "Canada"[t:53\%] \lor \text{birth}_\text{region}(x) = "foreign"[t:47\%].
\]
Mining Class Comparisons

**Comparison:** Comparing two or more classes

**Method:**
- Partition the set of relevant data into the target class and the contrasting class(es)
- Generalize both classes to the same high level concepts
- Compare tuples with the same high level descriptions
- Present for every tuple its description and two measures
  - support - distribution within single class
  - comparison - distribution between classes
- Highlight the tuples with strong discriminant features

**Relevance Analysis:**
- Find attributes (features) which best distinguish different classes
Concept Description vs. Cube-Based OLAP

- **Similarity:**
  - Data generalization
  - Presentation of data summarization at multiple levels of abstraction
  - Interactive drilling, pivoting, slicing and dicing

- **Differences:**
  - OLAP has systematic preprocessing, query independent, and can drill down to rather low level
  - AOI has automated desired level allocation, and may perform dimension relevance analysis/ranking when there are many relevant dimensions
  - AOI works on the data which are not in relational forms
Multidimensional Data Analysis in Cube Space

- Prediction Cubes: Data Mining in Multi-Dimensional Cube Space
- Multi-Feature Cubes: Complex Aggregation at Multiple Granularities
- Discovery-Driven Exploration of Data Cubes
Data Mining in Cube Space

- Data cube greatly increases the analysis bandwidth
- Four ways to interact OLAP-styled analysis and data mining
  - Using cube space to define data space for mining
  - Using OLAP queries to generate features and targets for mining, e.g., multi-feature cube
  - Using data-mining models as building blocks in a multi-step mining process, e.g., prediction cube
  - Using data-cube computation techniques to speed up repeated model construction
    - Cube-space data mining may require building a model for each candidate data space
    - Sharing computation across model-construction for different candidates may lead to efficient mining
Prediction Cubes

- **Prediction cube**: A cube structure that stores prediction models in multidimensional data space and supports prediction in OLAP manner
- Prediction models are used as building blocks to define the interestingness of subsets of data, i.e., to answer which subsets of data indicate better prediction
How to Determine the Prediction Power of an Attribute?

- Ex. A customer table $D$:
  - Two dimensions $Z$: $Time\ (Month,\ Year)$ and $Location\ (State,\ Country)$
  - Two features $X$: $Gender$ and $Salary$
  - One class-label attribute $Y$: $Valued\ Customer$

- Q: “Are there times and locations in which the value of a customer depended greatly on the customers gender (i.e., Gender: predictiveness attribute $V$)?”

- Idea:
  - Compute the difference between the model built on that using $X$ to predict $Y$ and that built on using $X - V$ to predict $Y$
  - If the difference is large, $V$ must play an important role at predicting $Y$
Efficient Computation of Prediction Cubes

- Naïve method: Fully materialize the prediction cube, i.e., exhaustively build models and evaluate them for each cell and for each granularity
- Better approach: Explore score function decomposition that reduces prediction cube computation to data cube computation
Chapter 4: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Data Warehouse Implementation
- Data Generalization by Attribute-Oriented Induction
- Summary
Chapter 3: Data Warehousing and OLAP Technology: An Overview

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

Summary
Summary: Data Warehouse and OLAP Technology

- Why data warehousing?
- A multi-dimensional model of a data warehouse
  - Star schema, snowflake schema, fact constellations
  - A data cube consists of dimensions & measures
- OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data warehouse architecture
- OLAP servers: ROLAP, MOLAP, HOLAP
- Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
- From OLAP to OLAM (on-line analytical mining)