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

- **time**
  - time_key
  - day
  - day_of_the_week
  - month
  - quarter
  - year

- **branch**
  - branch_key
  - branch_name
  - branch_type

- **location**
  - location_key
  - street
  - city
  - state_or_province
  - country

- **item**
  - item_key
  - item_name
  - brand
  - type
  - supplier_type

- **Sales Fact Table**
  - time_key
  - item_key
  - branch_key
  - location_key
  - units_sold
  - dollars_sold
  - avg_sales
Example of Snowflake Schema

- **time**
  - time_key
  - day
  - day_of_the_week
  - month
  - quarter
  - year

- **location**
  - location_key
  - street
  - city_key

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

- **item**
  - item_key
  - item_name
  - brand
  - type
  - supplier_key

- **supplier**
  - supplier_key
  - supplier_type

- **branch**
  - branch_key
  - branch_name
  - branch_type

- **location**
  - location_key
  - street
  - city_key

- **city**
  - city_key
  - city
  - state_or_province
  - country
Example of Fact Constellation

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

**Measures**
- **branch_key**
- **branch_name**
- **branch_type**

**Item**
- **item_key**
- **item_name**
- **brand**
- **type**
- **supplier_type**

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

**Location**
- **location_key**
- **street**
- **city**
- **province_or_state**
- **country**
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

```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)
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

Region

Product

Month

Industry  Region  Year
Category  Country  Quarter
Product  City  Office  Week
Month  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
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

Extract Transform Load Refresh

Metadata

Monitor & Integrator

OLAP Server

Data Warehouse

Serve

Analysis Query Reports Data mining

OLAP Engine

Front-End Tools

Data Marts
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

Multi-Tier Data Warehouse

Data Mart

Data Mart

Enterprise Data Warehouse

Model refinement

Model refinement

Define a high-level corporate data model
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 middleware
  - 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 the following $2^3$ Group-Bys (For n dim, $2^n$ Group-Bys)
  
  ```
  (date, product, customer), 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 Alls

- $C_i$ is cardinality of i attribute, then cardinality of cube = $(C_1 + 1) * (C_2 + 1) * ... * (C_N + 1)$ 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

```sql
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>
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
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 \( \geq 2 \)?
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
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)