Data Warehouses

Yanlei Diao

Slides Courtesy of R. Ramakrishnan and J. Gehrke
Introduction

- In the late 80s and early 90s, companies began to use their DBMSs for complex, interactive, exploratory analysis of historical data.

Operational Data
(purchase transactions: store, customer, products, sales, etc.)

Decision Making:
- how much of which products to order for which store?
- when to deliver the products?
- benefits of promotional offers?
Data Warehousing

- **Data:**
  - Integrated data spanning long time periods, often augmented with summary information.
  - Large volumes: several terabytes to petabytes common.

- **Queries:**
  - Interactive response time expected for complex queries.
  - Ad-hoc updates uncommon.
A data warehouse (DW) is an organization-wide data repository, used for decision making

- An integrated enterprise warehouse collects info about all subjects, e.g. customers, products, sales, assets, personnel.
- The data is used to assist in decision making
  - e.g., how much of which products to order for which stores, when to deliver the products, the benefits of various promotional offers, etc.
- Analytics is called On-Line Analytic Processing (OLAP).
- OLAP tasks slowed down the normal operation of the company, called On-Line Transaction Processing (OLTP), leading to separation of DWs from operational DBs.
## OLTP vs OLAP Databases

<table>
<thead>
<tr>
<th>OLTP / Operational / Production</th>
<th>OLAP / Data Warehouse / DSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operate the business / Clerks</td>
<td>Diagnose the business / Managers</td>
</tr>
<tr>
<td>Short queries, small amts of data</td>
<td>Large queries, large amts of data</td>
</tr>
<tr>
<td>Current data</td>
<td>Current and historical data</td>
</tr>
<tr>
<td>Queries change data</td>
<td>Queries are mostly read-only</td>
</tr>
<tr>
<td>• Examples: customer inquiry, order entry</td>
<td>• OLAP, data mining, statistics, visualization, etc.</td>
</tr>
<tr>
<td>Legacy applications, heterogeneous databases</td>
<td>Opposite</td>
</tr>
<tr>
<td>Often distributed</td>
<td>Often integrated and centralized (Warehouse)</td>
</tr>
</tbody>
</table>
Overview of Topics

1. Introduction
   - Operational vs. Warehouse

2. Multidimensional Data
   - Data model & schema

3. Queries
   - OLAP Queries
   - CUBE Operator
   - Window Operator

4. Implementation Algorithms
   - Bitmap Index
   - MOLAP vs ROLAP

5. Materialized Views
   - View definition
   - Query answering using views
   - View selection

6. Constructing a Data Warehouse (ETL)
2. Multidimensional Data (Ch 25.2)

- To support OLAP, warehouse data is often structured multidimensionally, as measures and dimensions.
  - **Measure**: numeric attribute, e.g. sales amount
  - **Dimension**: attribute categorizing the measure, e.g. product, store, date of sale.

- **Star schema**:
  - The central *fact table* contains a foreign key for each dimension, plus an attribute for each measure.
  - There will also be a *dimension table* for each dimension.
This kind of schema, called a **star schema**, is very common in OLAP applications.

Is this a good design of the schema?
Dimension Hierarchies

- For each dimension, some of the attributes may be organized in a hierarchy:

  **PRODUCT**
  - category
  - pname
  - PID

  **TIME**
  - year
  - quarter
  - week
  - date

  **LOCATION**
  - state
  - city
  - ZIP
Star/Snowflake Schemas

**Star schema**
- **Order**
  - OrderNo
  - OrderDate
- **Customer**
  - CustomerNo
  - CustomerName
  - CustomerAddress
  - City
- **Salesperson**
  - SalespersonID
  - SalespersonName
  - City
  - Quota

**Fact Table**
- OrderNo
- CustomerNo
- SalespersonID
- ProdNo
- DateKey
- CityName
- Quantity
- TotalPrice

**Product**
- ProdNo
- ProductName
- ProductDescr
- Category
- CategoryDescr
- UnitPrice

**Date**
- DateKey
- Date
- Month
- Year

**Snowflake schema**
- **Order**
  - OrderNo
  - OrderDate
- **Customer**
  - CustomerNo
  - CustomerName
  - CustomerAddress
  - City
- **Salesperson**
  - SalespersonID
  - SalespersonName
  - City
  - Quota

**Fact Table**
- OrderNo
- CustomerNo
- SalespersonID
- ProdNo
- DateKey
- CityName
- Quantity
- TotalPrice

**Product**
- ProdNo
- ProductName
- ProductDescr
- Category
- CategoryDescr
- UnitPrice

**Date**
- DateKey
- Date
- Month
- Year

**Category**
- CategoryName
- CategoryDescr

**City**
- CityName
- State
- Country

**Month**
- Month
- Year

**State**
- State
- Country
Star/Snowflake Schemas

- Why normalize?
  - Save space
  - Remove store redundancy and anomalies
  - If fully normalized, it is a snowflake schema

- Why denormalize?
  - Performance benefits, e.g., avoiding joins

- Which is more important in Data Warehouses?
Examples of Multi-Dimensional Data

- **Purchase (ProductID, StoreID, DateID, Amt)**
  - Product(ID, SKU, size, brand)
  - Store(ID, Address, Sales District, Region, Manager)
  - Date (ID, Week, Month, Holiday, Promotion)

- **Claims (ProvID, MembID, ProcedureID, DateID, Cost)**
  - Providers(ID, Practice, Address, ZIP, City, State)
  - Members(ID, Contract, Name, Address)
  - Procedure (ID, Name, Type)

- **Telecomm (CustID, SalesRepID, ServiceID, DateID)**
  - SalesRep(ID, Address, Sales District, Region, Manager)
  - Service(ID, Name, Category)
  - ...
MOLAP vs ROLAP

- Multidimensional data can be stored physically in a (disk-resident, persistent) array; called MOLAP systems.
- Or, data can be stored as a relation; called ROLAP systems.
- ROLAP:
  - The main relation, which relates dimensions to a measure (e.g., sales), is the fact table.
  - Each dimension has additional attributes in a dimension table.
  - E.g., Products(pid, locid, timeid, amt)
  - Fact tables are much larger than dimensional tables.
Multidimensional Data Model

- Collection of numeric measures, which depend on a set of dimensions.
  - E.g., measure Amt, dimensions Product (key: pid), Time (timeid), and Location (locid).

Slice locid=1

<table>
<thead>
<tr>
<th>pid</th>
<th>timeid</th>
<th>locid</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>35</td>
</tr>
</tbody>
</table>
3. OLAP Queries (Ch 25.3)

- Influenced both by SQL and by spreadsheets.
- A common operation is to **aggregate** a measure over one or more dimensions.
  - Find total sales.
  - Find total sales for each city, or for each state.
  - Find top five products ranked by total sales.
OLAP Queries

- **Roll-up**: aggregates at increasingly coarser levels of a dimension hierarchy.
  - E.g., Given total sales by ZIP, we can roll-up to get sales by city, and then by state.
OLAP Queries

- **Drill-down**: the inverse of roll-up.
  - E.g., Given total sales by state, can drill-down to compute total sales by city.
  - E.g., Can also drill-down on a different dimension to compute total sales of each state by product.

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>category</td>
</tr>
<tr>
<td>city</td>
<td>pname</td>
</tr>
<tr>
<td>ZIP</td>
<td>PID</td>
</tr>
</tbody>
</table>
OLAP Queries

- **Pivoting**: aggregates on selected dimensions
  - E.g., Pivoting on State and Year yields the **cross-tabulation** as shown below

<table>
<thead>
<tr>
<th>Year</th>
<th>OR</th>
<th>CA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>63</td>
<td>81</td>
<td>144</td>
</tr>
<tr>
<td>2008</td>
<td>38</td>
<td>107</td>
<td>145</td>
</tr>
<tr>
<td>2009</td>
<td>75</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td>Total</td>
<td>176</td>
<td>223</td>
<td>339</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>pid</th>
<th>timeid</th>
<th>locid</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>35</td>
</tr>
</tbody>
</table>
OLAP Queries

- **Slicing and Dicing:** equality and range selections on one or more dimensions.
Tableau Demo

- https://www.youtube.com/watch?v=pXYgsd9xOZI

- Note the many measures.
- Pivot sales on category and region.
- Clear date, pivot on product and drill down on subcategory.
- Add profit as another measure
- Change bars to circles
- Pivot on dates (columns)
Comparison with SQL Queries

- The cross-tabulation obtained by pivoting:

<table>
<thead>
<tr>
<th>pid</th>
<th>timeid</th>
<th>locid</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>CA</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>81</td>
<td>144</td>
</tr>
<tr>
<td>38</td>
<td>107</td>
<td>145</td>
</tr>
<tr>
<td>75</td>
<td>35</td>
<td>110</td>
</tr>
<tr>
<td>176</td>
<td>223</td>
<td>339</td>
</tr>
</tbody>
</table>
Comparison with SQL Queries

- The cross-tabulation obtained by pivoting can also be computed using a collection of SQL queries:

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

SELECT T.year, SUM(S.amt)
FROM Sales S, Times T
WHERE S.timeid=T.timeid
GROUP BY T.year

SELECT L.state, SUM(S.amt)
FROM Sales S, Location L
WHERE S.locid=L.locid
GROUP BY L.state
```
The CUBE Operator

- If there are $k$ dimensions, we have $2^k$ possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions.

- **GROUP BY CUBE**(pid, locid, timeid)
  - Equivalent to rolling up Sales on all eight subsets of the set \{pid, locid, timeid\}

```sql
SELECT grouping-list, SUM(S.amt)
FROM   Sales S
GROUP BY CUBE(grouping-list)
ORDER BY grouping-list
```
Cube Operator (cont’d)

- GROUP BY CUBE(pid, locid, timeid) -- SUM Sales
  - Equivalent to rolling up Sales on all eight subsets of \{pid, locid, timeid\}; each roll-up amounts to a query of the form:
    
    ```
    SELECT SUM(S.amt)
    FROM Sales S
    GROUP BY grouping-list
    ```

- The lattice of group-by operations

```
<table>
<thead>
<tr>
<th>pid, locid, timeid</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid, locid</td>
</tr>
<tr>
<td>pid, timeid</td>
</tr>
<tr>
<td>locid</td>
</tr>
<tr>
<td>locid, timeid</td>
</tr>
<tr>
<td>pid</td>
</tr>
<tr>
<td>timeid</td>
</tr>
<tr>
<td>ALL</td>
</tr>
</tbody>
</table>
```
Overview of Topics

1. Introduction
   - Operational vs. Warehouse

2. Multidimensional Data
   - Data model & schema

3. Queries
   - OLAP Queries
   - CUBE Operator
   - Window Operator

4. Implementation Algorithms
   - Bitmap Index
   - MOLAP vs ROLAP

5. Materialized Views
   - Query answering using views
   - View maintenance
   - View selection

6. Constructing a Data Warehouse (ETL)

7. An example of data mining
   - Association rule mining
5. Views and Decision Support (25.8,9)

- In large databases, *precomputation* is necessary for fast response times
  - Examples: brain, google
- Example: Precompute daily sums for the cube.
  - What can be derived from those precomputations?
- These precomputed queries are called *materialized views*
  - Another name: Indexed views in SQL Server.
Materialized View Example

**Materialized View**

```sql
CREATE VIEW DailySum(date, sumamt) AS
    SELECT date, SUM(amt)
    FROM Times Join Sales USING(timeid)
    GROUP BY date
```

**Query**

```sql
SELECT week, SUM(amt)
FROM Times Join Sales USING(timeid)
GROUP BY week
```

**Modified Query**

```sql
SELECT week, SUM(sumamt)
FROM Times Join DailySum USING (date)
GROUP BY week
```
Views

- A view is a named query

- A materialized view is the stored result of a query

Why are views interesting?
Views

- A view is a named query

- A materialized view is the stored result of a query

Why are views interesting?

- Query optimization

- Independence of the physical and the logical views

- Access rights management
Using a View for a Query

Query:

```
Select  Advises.prof, Advises.student, Registered.quarter
From    Registered, Teaches, Advises
Where   Registered.c-number = Teaches.c-number and
        Registered.quarter = Teaches.quarter and
        Advises.prof = Teaches.prof and Advises.student =
        Registered.student and Registered.quarter >= "winter98"
```

V1:

```
Select  Registered.student, Teaches.prof, Registered.quarter
From    Registered, Teaches
Where   Registered.c-number = Teaches.c-number and
        Registered.quarter = Teaches.quarter and
        Registered.quarter > "winter97".
```
- Query:

Query: 

\[ \text{Quarter} \geq \text{"winter 98"} \]

- V1

Query: 

\[ \text{Quarter} > \text{"winter 97"} \]
Query:

Quarter >= "winter 98"

V2

Quarter >= "winter 98"
Query:
Quarter >= “winter 98”

V3
Quarter >= “winter 98”