1. **What is a data warehouse?**

2. A multi-dimensional data model

3. Data warehouse architecture

4. Data warehouse implementation

5. Further development of data cube technology

6. Data warehousing and data mining
What is Data Warehouse?

• A decision support database that is maintained separately from the organization’s operational database. Not rigorous.

• “A data warehouse is a subject-oriented, integrated, time-variant, and non volatile collection of data in support of management’s decision-making process.” W. H. Inmon

Data Warehouse - Subject-Oriented

• Organized around major subjects, such as product, sales.

• Focusing on the modelling 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 process.
Data Warehouse - Integrated

• Constructed by integrating multiple, heterogeneous data sources
  – relational databases, flat files, …

• Data cleaning and data integration techniques are applied.
  – Ensure consistency in naming conventions (e.g., LastName and FamilyName in DB1 and DB2 have the same signification)
  – encoding structures (e.g, Attribute User_Id is a long int in DB1 and it is a string in DB2
  – attribute measures (e.g, cm vs inch) …
  – When data is moved to the warehouse, it is converted.

Data Warehouse - Time Variant

• Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
  – Operational database: current value data.

• Every data in the data warehouse
  – Contains an element of time, explicitly or implicitly
  – But the data of operational database may or may not contain “time element”.
Data Warehouse - Non-Volatile

- A physically separate store.
- 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 querying (read)

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration:
  - Build wrappers/mediators on top of heterogeneous databases
  - Query driven approach
    - A query posed to a client site, will be transformed into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set

- Data warehouse: update-driven
  - 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, …

• OLAP (On-Line Analytical Processing)
  - Major task of data warehouse system
  - Data analysis and decision making

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>users</td>
<td>Any one</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, detailed,</td>
<td>historical, summarized,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>multidimensional integrated,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>consolidated</td>
</tr>
<tr>
<td>access</td>
<td>read/write</td>
<td>lots of scans</td>
</tr>
<tr>
<td></td>
<td>index/hash 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>DB size</td>
<td>100MB-GB</td>
<td>100GB-TB</td>
</tr>
<tr>
<td>metric</td>
<td>transaction throughput</td>
<td>query throughput, response</td>
</tr>
</tbody>
</table>
From tables 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, type,...)
    - time(day, week, month, quarter, year)
    - location(location_name, country)
  - **Fact table** contains **measures** and **keys** to related dimension tables
From tables to Data Cubes

2-D view of sales cross-tabulation (pivot table)

<table>
<thead>
<tr>
<th>Location = « Mekkah »</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>item</strong></td>
</tr>
<tr>
<td><strong>time</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

The symbol * means ALL

Relational representation of pivot table

By considering the pivot tables of Mekkah, Madimah and Quds we obtain the following cube
From tables to Data Cubes

3-D view of sales cube

Cube: A lattice of Cuboids

0-D (apex) cuboid

1-D cuboids

2-D cuboids

3-D (base) cuboid
Cube: A lattice of Cuboids

- How many cuboids in an \( n \)-dimensional cube with \( L \) levels?

\[
T = \prod_{i=1}^{n} (L_i + 1)
\]

- If \( n=10 \) and each dimension has one level then
  \( T = (2)^{10} \)
- If \( n=10 \) and each dimension has 4 levels then
  \( T = (4+1)^{10} \)
Relational Data Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  1. **Star schema**: A fact table in the middle connected to a set of dimension tables
  2. **Snowflake schema**: represents dimensional hierarchy by normalizing the dimension tables (i.e., each level of a dimension represented in one table)
     - save storage
     - reduces the effectiveness of browsing
  3. **Fact constellations**: Multiple fact tables share dimension tables

Example of Star Schema
Example of Snowflake Schema

This is not a full snowflake schema

Example of Fact Constellation
Defining a Star Schema in DMQL

```dmql
define cube sales_star [time, item, branch, location]:
currency_sold = sum(sales_price),
avg_sales = avg(sales_price),
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 a Snowflake Schema in DMQL

```dmql
define cube sales_snowflake [time, item, branch, location]:
currency_sold = sum(sales_price),
avg_sales = avg(sales_price),
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))
```
Multidimensional Data

- Dimensions are: product, month, region
- Measure is sales_amount

Hierarchical summarization paths

Industry Region Year
Category Country Quarter
Product City Month Week
Office Day

An example of Data Cube

Total annual sales of TV in Mekkah

Mekkah
Medinah
Quds

TV PC DVD
1Qtr 2Qtr 3Qtr 4Qtr sum

Product Date
sum

All, All, All
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.

OLAP operations and SQL

*Simple star schema*

- **time**
  - time_key
  - month
  - quarter
  - year
- **product**
  - item_key
  - name
  - type
- **location**
  - location_key
  - city
  - state
  - country
- **Sales Fact Table**
  - time_key
  - item_key
  - location_key
  - qsales
Creation of tables

- create materialized view `product` as
  
  ```sql
  select name, type
  from 'DB1'.product
  ```

- create materialized view `location` as
  
  ```sql
  select city, state, country
  from 'DB2'.location
  ```

- The `time` dimension can be created in the same manner

Data Cube

```
<table>
<thead>
<tr>
<th>Product</th>
<th>time</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>PC</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>DVD</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>sum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Total annual sales of TV in Mekkah**

```
<table>
<thead>
<tr>
<th>Product</th>
<th>time</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>PC</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>DVD</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td>sum</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

IS 466 – Data Warehousing - Dr. Mourad Ykhlef
Roll up query

```sql
select quarter, year, item_key, location_key, sum(qsales)
from sales S, time T
where S.time_key = T.time_key
group by year, quarter, item_key, location_key
```

- In order to provide three dimensions we have to concatenate quarter and year in one string.

Drill down query

- Consider that roll up query is created as follows

```sql
create view salesQuarter as
  select quarter, year, item_key, location_key, sum(q_sales)
  from sales S, time T
  where S.time_key = T.time_key
  group by year, quarter, item_key, location_key
```

- Drill down(quarter⇒month)
  - Use of salesQuarter, sales (provides qSales) and time dimension
Slice query

create view sliceView as
select time_key, item_key, location_key, q_sales
from sales S
where S.time_key = 1000
order by item_key, location_key

Dice query

select time_key, item_key, location_key, q_sales
from sliceView
where sliceView.location_key = 2000

Dont forget order by
1. What is a data warehouse?
2. A multi-dimensional data model
3. **Data warehouse architecture**
4. Data warehouse implementation
5. Further development of data cube technology
6. Data warehousing and data mining

Three Data Warehouse Models

- **Enterprise warehouse**: collects all information about subjects (*customer, products, sales, assets, personnel*) that span the entire organization
  - Requires extensive business modeling
  - May take years to design and build

- **Data Mart**: Departmental subsets that focus on selected subjects: *Marketing data mart: customer, product, sales*
  - Faster roll-out
  - Complex integration in the long term

- **Virtual warehouse**
  1. A set of views over operational databases
  2. Only some of views may be materialized
OLAP Server Architectures

• **Relational OLAP (ROLAP):**
  - Leaves detail values in the relational fact table
  - Stores aggregated values in the relational database as well.

• **Multidimensional OLAP (MOLAP)**
  - Stores both detail and aggregated within the cube.

• **Hybrid OLAP (HOLAP):**
  - Leaves detail values in the relational fact table
  - Stores aggregated values in the cube.
1. What is a data warehouse?
2. A multi-dimensional data model
3. Data warehouse architecture
4. **Data warehouse implementation**
5. Further development of data cube technology
6. Data warehousing and data mining

**Efficient Data Cube Computation**

- Data cube can be viewed as a lattice of cuboids
  1. The bottom-most cuboid is the base cuboid
  2. The top-most cuboid (apex) contains only one cell
  3. 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
  1. Based on size, sharing, access frequency, etc.
Cube Operation

- Cube definition and computation in DMQL
  
  Define cube sales[item, city, year]: sum(sales_in_currency)
  
  compute cube sales

- Transform it into a SQL-like language (with a new operator cube by -Gray et al.’96-)
  
  SELECT item, city, year, SUM (sales_in_currency)
  FROM SALES
  CUBE BY item, city, year

- Need compute the following Group-Bys
  
  (item, city, year),
  (item, city), (item, year), (city, year),
  (item), (city), (year)

Partial materialization (choose view to answer a query)

- Identify all of the materialized cuboids that may potentially be used to answer a query.

- Pruning the above set using knowledge of dominance relationship among cuboids.

- Estimating the costs of using the remaining materialized cuboids and selecting the cuboid with the least cost.
Partial materialization
(choose view to answer a query)

Define cube sales[year, product, location]: sum(sales_in_currency)

Dimension hierarchies used are
  item < brand
  street < city < state < country

Cuboids are:
  cuboid 1: {item, city, year}
  cuboid 2: {brand, country, year}
  cuboid 3: {brand, state, year}
  cuboid 4: {item, state} where year = 2000

Query {brand, state} with year = 2000?

cuboid 1 costs the most since item and city are at lower level than brand and state
cuboid 2 can not be used since state < country
Partial materialization
(choose view to answer a query)

Define cube sales[year, product, location]: sum(sales_in_currency)

Dimension hierarchies used are
- item < brand
- street < city < state < country

Cuboids are:
- cuboid 3: {brand, state, year}
- cuboid 4: {item, state} where year = 2000

Query {brand, state} with year = 2000?

cuboid 3 is better than cuboid 4 if
- there are few year values associated with items
- and several items for each brand

cuboid 4 is better than cuboid 3 if efficient indices are available for cuboid 4.

Full materialization
Multi-way Array Aggregation for Cube Computation

- Partition arrays into chunks (a small subcube which fits in memory).
- Compute aggregates in “multiway” by visiting cube cells in the order (1) which minimizes the # of times to visit each cell, and (2) reduces memory access and storage cost.

What is the best traversing order to do multi-way aggregation?
Full materialization
Multi-way Array Aggregation for Cube Computation

- Cuboids are: A, B, C, AB, BC, AC, ABC and Ø
- size(A) = 40, size(B)=400, size(C)=4000
- The size of each chunk of A, B and C are 10, 100 and 1000
- Computation of cuboid BC is done by computing cuboids b<sub>i</sub>c<sub>j</sub>.
- b<sub>0</sub>c<sub>0</sub> is fully aggregated after scanning 1,2,3,4.
- For BC aggregation we scan chunk 1 to 64
- Is the computation of cuboids AC et AB, needs scanning of 64 chunks another time? NO
  1. Scanning of a<sub>0</sub>b<sub>0</sub>c<sub>0</sub> leads to compute a<sub>0</sub>b<sub>0</sub> and a<sub>0</sub>c<sub>0</sub> (Multi-way aggregation)
  2. So by scanning of 64 chunks one time, one can compute 3 cuboides AB, AC and BC
Full materialization
Multi-way Array Aggregation for Cube Computation

• In order to avoid bringing 3-D chunk into memory more than once the minimum memory requirement for holding 2-D plans according to chunk ordering of 1 to 64 is
  \[40 \times 400 \text{ (for AB)} + 40 \times 1000 \text{ (for one row of AC)} \]
  \[+ 100 \times 1000 \text{ (for one chunk of BC)} = 156000\]

• If the chunk ordering is 1,17,33,49,5,21,37,53,…the memory requirement is
  \[400 \times 4000 \text{ (for BC)} + 10 \times 4000 \text{ (for one row of CA)} \]
  \[+ 10 \times 100 \text{ (for one chunk of AB)} = 1641000\]

• Limitation of the method: computing well only for a small number of dimensions
  – If there are a large number of dimensions, "bottom-up computation" (Beyer & Ramakrishnan, SIGMOD'99) and "iceberg cube computation" methods can be used.
No materialization
Indexing OLAP Data

Bitmap Index

- An effective indexing technique for attributes with low-cardinality domains.

- Each value in the column has a bit vector

- The length of the bit vector: # of records in the base table

- Example: attribute gender has value M and F. A table of million people needs 2 lists of million bits.

### Bitmap Index

<table>
<thead>
<tr>
<th>Cust</th>
<th>Region</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>N</td>
<td>H</td>
</tr>
<tr>
<td>C2</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>C3</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C4</td>
<td>W</td>
<td>H</td>
</tr>
<tr>
<td>C5</td>
<td>S</td>
<td>L</td>
</tr>
<tr>
<td>C6</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C7</td>
<td>W</td>
<td>H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region Index</th>
<th>Rating Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowId</td>
<td>N</td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

**select** cust  
**from** base table  
**where** region=“W” and rating=“L”
### Bitmap Index

**base table**

<table>
<thead>
<tr>
<th>Cust</th>
<th>Region</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>N</td>
<td>H</td>
</tr>
<tr>
<td>C2</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>C3</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C4</td>
<td>W</td>
<td>H</td>
</tr>
<tr>
<td>C5</td>
<td>S</td>
<td>L</td>
</tr>
<tr>
<td>C6</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C7</td>
<td>W</td>
<td>H</td>
</tr>
</tbody>
</table>

**Region Index**

<table>
<thead>
<tr>
<th>RowId</th>
<th>N</th>
<th>S</th>
<th>E</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Rating Index**

<table>
<thead>
<tr>
<th>RowId</th>
<th>H</th>
<th>M</th>
<th>L</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>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Relations: Region and Rating**

- region = “W”
- rating = “L”

---

### Bitmap Index

**base table**

<table>
<thead>
<tr>
<th>Cust</th>
<th>Region</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>N</td>
<td>H</td>
</tr>
<tr>
<td>C2</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>C3</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C4</td>
<td>W</td>
<td>H</td>
</tr>
<tr>
<td>C5</td>
<td>S</td>
<td>L</td>
</tr>
<tr>
<td>C6</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>C7</td>
<td>W</td>
<td>H</td>
</tr>
</tbody>
</table>

**Region Index**

<table>
<thead>
<tr>
<th>RowId</th>
<th>N</th>
<th>S</th>
<th>E</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Rating Index**

<table>
<thead>
<tr>
<th>RowId</th>
<th>H</th>
<th>M</th>
<th>L</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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Relations: Region and Rating**

- region = “W”
- rating = “L”

---

IS 466 – Data Warehousing - Dr. Mourad Ykhlef
Indexing OLAP Data: Join Indices

- Join index roughly $JI(C_f, Row-id)$ where

  $$D(C_d, \ldots) \triangleright C_d=C_f \quad F(Row-id, C_f, \ldots)$$

- Traditional indices map the values to a list of record ids

- In data warehouse, join index relates the values of the dimensions of a star schema to rows (ids) in the fact table.

- Join indices can span multiple dimensions

Join Index

<table>
<thead>
<tr>
<th>Key</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mekkah</td>
</tr>
<tr>
<td>2</td>
<td>Medina</td>
</tr>
<tr>
<td>3</td>
<td>Quds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC</td>
</tr>
<tr>
<td>2</td>
<td>TV</td>
</tr>
<tr>
<td>3</td>
<td>DVD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Location

Sales

<table>
<thead>
<tr>
<th>Lkey</th>
<th>Pkey</th>
<th>Tkey</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>rid4</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>rid5</td>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>rid6</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>rid7</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>rid8</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>rid9</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>rid10</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>rid11</td>
<td>3</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>rid12</td>
<td>3</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>rid13</td>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>rid14</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>rid15</td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>rid16</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>rid17</td>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>rid18</td>
<td>2</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>rid19</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>rid20</td>
<td>3</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>rid21</td>
<td>3</td>
<td>3</td>
<td>30</td>
</tr>
</tbody>
</table>

Location-ProductJI

<table>
<thead>
<tr>
<th>CityK</th>
<th>PrdK</th>
<th>Rid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>rid4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>rid13</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>rid5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>rid14</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>rid6</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>rid15</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Join Index

<table>
<thead>
<tr>
<th>Key</th>
<th>City</th>
<th>Lkey</th>
<th>Pkey</th>
<th>Tkey</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mekkah</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Medinah</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Quds</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

### Sales

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
<th>Lkey</th>
<th>Pkey</th>
<th>Tkey</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>TV</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>DVD</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

### LocationJI

<table>
<thead>
<tr>
<th>CityK</th>
<th>Rid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rid4</td>
</tr>
<tr>
<td>1</td>
<td>rid5</td>
</tr>
<tr>
<td>1</td>
<td>rid6</td>
</tr>
<tr>
<td>1</td>
<td>rid7</td>
</tr>
<tr>
<td>1</td>
<td>rid8</td>
</tr>
<tr>
<td>1</td>
<td>rid9</td>
</tr>
</tbody>
</table>

### Time

<table>
<thead>
<tr>
<th>Key</th>
<th>Month</th>
<th>Lkey</th>
<th>Pkey</th>
<th>Tkey</th>
<th>Qnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

### Efficient Processing of 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 to which materialized cuboid(s) the relevant operations should be applied.
- Exploring indexing structures and compressed vs. dense array structures in MOLAP
Data Warehouse Utilities

- **Data extraction:**
  - get data from sources
- **Data cleaning:**
  - detect errors in the data and rectify them when possible
- **Data transformation:**
  - convert data from 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

1. What is a data warehouse?
2. A multi-dimensional data model
3. Data warehouse architecture
4. Data warehouse implementation
5. **Further development of data cube technology**
6. Data warehousing and data mining
Complex Aggregation at Multiple Granularities: Multi-Feature Cubes

- Multi-feature cubes (Ross, et al. 1998): Compute complex queries involving multiple dependent aggregates at multiple granularities
- Ex. Grouping by all subsets of \{item, region, month\}, find the maximum price in 2000 for each group, and the total sales among all maximum price tuples

```sql
select item, region, month, max(price), sum(R.sales)
from sales
where year = 2000
cube by item, region, month: R
such that R.price = max(price)
```

1. What is a data warehouse?
2. A multi-dimensional data model
3. Data warehouse architecture
4. Data warehouse implementation
5. Further development of data cube technology
6. Data warehousing and data mining
From On-Line Analytical Processing to On-Line Analytical Mining (OLAM)

- OLAM integrates OLAP with Data mining
- Why OLAM?
  - 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.
Summary

- 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
- OLAP servers: ROLAP, MOLAP, HOLAP
- Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Multiway array aggregation
  - Bitmap index and join index implementations
- Further development of data cube technology
- From OLAP to OLAM (on-line analytical mining)

A1. Views and Warehousing

- A DW is just a collection of asynchronously replicated tables and periodically maintained views

- OLAP queries are typically aggregate queries.

- Analysts want fast answers to OLAP queries over very large datasets and it is natural to consider precomputing views.

- **CUBE** operator gives rise to several aggregate queries.
**A1. Views and Warehousing (Virtual view)**

create view *RegionalSales* (category, sales, state) as
select  P.category, S.sales, L.state
     from  Products P, Sales S, Location L
    where  P.pid = S.pid and S.locid=L.locid

Compute the total sales for each category by state?

    select R.category, R.state, sum(R.sales)
         from *RegionalSales*  R
        group by R.category, R.state

• **Query unfolding** consists to replace the occurrence of RegionalSales in the query by view definition

**A1. Views and Warehousing (Materialised view)**

• Unfolding is not suitable for OLAP because it is a time consumer.

• View materialization is better: there is no necessity to unfold, the query is executed directly on the pre-computed result.
  - The drawback is that we must maintain the consistency of the materialized view whenever the underlying tables are updated.

• For more details on data warehousing see the site
  http://www.ondelette.com/OLAP/dwbib.html

End