Spring 2009	**CSC 466:** Knowledge Discovery from Data	Alexander Dekhtyar

Data Warehousing and On-line Analytical Processing (OLAP)

Data Warehouses

Data Warehouse: *"a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process"*[1]

Key properties:

- Subject-oriented;
- Integrated: assembled from various *heterogenous* sources;
- Time-variant: historic perspective;
- Nonvolatile: data warehouses are separate from operational databases.

Data Warehouses vs. Relational Databases

Question: *Why Data Warehouses are different from regular relational databases that support business operations?*

There are really many aspects to consider.

Purpose: Databases: support day-to-day operations of business entities (customer-oriented);
Data Warehouses: support strategic planning and decision-making (market-oriented).

Contents: Databases: low granularity, transactions;
Data Warehouses: high granularity, summaries, aggregates.

Queries: Databases: short transactions; select-where-from;
Data Warehouses: aggregate, summarize, find patterns (mine data).

Access: Databases: transaction processing, concurrency control, distributivity, crash recovery, read-write, *many users*;
Data Warehouses: mostly read, *few users*.

Metric: Databases: efficiency of query processing;
Data Warehouse: quality of answer;

To summarize:

- Databases: low-granularity data, many users, simple, efficient queries (always with exact answer);
- Data Warehouses: complex data, few users, complex queries (exact answers not always exist).

OLTP: On-line Transaction Processing: Databases;

OLAP: On-line Analytical Processing: Data Warehouses.

What data?

Data Warehouse is a technical term that typically refers to analytical databases build for automated *decision support* in business.

However, the theory of database models for data warehouses is applicable to a wider range of applications.

- **Operational Transactional Databases.** Purchases, orders, etc... *The typical source of data warehouses.*
- **Scientific data.** Results of large experimental runs, where experimental design involves multiple independent variables.
- **Observational data.** Results of scientific (or non-scientific) observations.

Common feature: the original dataset contains records of individual transactions/experiments/observations. The data warehouse contains *aggregated information*.

Data cubes: A Multidimensional Data Model

data dimension: a (type of) *entity* or a *perspective* used to organize the data.

In relational databases constructed from E-R diagrams, relational tables represent either *entity sets* or *relationship sets*. E-R models try to construct only relationships between (typically) pairs of entity sets. Then, relational tables representing such relationship sets correspond to 2D data cubes: a mapping between two dimensions (specified by the entity sets that participate).

We can take this analogy further by specifying that 3- and higher dimensional data cubes correspond to relationship sets that connect more 3 or more different entity sets. Sometimes also, a dimension will not be a proper entity set: e.g., *time*.

Data Cubes. A **data cube** is a relational table that consists of two types of attributes:

1. Measure attributes. These attributes *measure* certain quantities within the data. They can be *aggregate* (e.g., number of items purchased) or non-aggregate (e.g., price).
2. Dimension attributes. These attributes specify the features over which the measures are established/computed.

Data cube tables also often are called fact tables.

Example. Consider a video-rentals chain that operates in some region. The chain consists of a number of locations. The *operational database* at each location contains a list of movies that the location has in stock, list of customers, and records for each video-rental transaction. Additionally, a list of all locations is also available.

```
Locations(Id, Street, City, State, Zip, County);
Movies(Id, Title, Year, Studio, Genre, RentalPrice);
Customers(MemberId, Name, Address, City, State, Zip, Phone);
InStock(Location, Movie, NumCopies)
Rentals(Location, Movie, Customer, RentalDate, Due, Returned);
```

A *data warehouse* for the video-rental company can contain[1] summarized information about the rentals for each individual movie by location (stored either as the number of rentals or the revenue generated by a specific title). This information can further be subdivided by *time*: the data is given for each quater.

We can create a data cube (fact) table based on the table of rentals by aggregating the customer information:

```
CREATE TABLE RentalCube3D AS
        ( SELECT Location, Movie, RentalDate, COUNT(*) AS NumRentals
          FROM Rentals
          GROUP BY Location, Movie, RentalDate
         )
;
```

The statement above generates a 3D data cube with the dimensions *Movie Title*, *Store Location* and *Time*.

Lattice of Cuboids

It is hard to visualize N-dimensional cubes with $N > 3$. One of the possible visualizations of N-dimensional data cube structure represented as a *lattice of cuboids*.

Given an N-dimensional data cube with dimensions $a_1, \ldots, a_N$ a *cuboid* is any subset $b_1, \ldots b_k$ of $a_1, \ldots, a_N$. A cuboid defines a *projection* of N dimensional data onto k dimensions. Cuboid $a_1, \ldots, a_N$ is called a *base cuboid*. $\emptyset$ is called an *apex cuboid*.

The *Lattics of Cuboids* is a graph G whose vertices are all subsets of $a_1, \ldots, a_N$ (cuboids). There is an edge from a node A to a node B is $A \subset B$. The Lattice of cuboids is typically shown in layers, with the apex cuboid at the top (or bottom) layer and the base cuboid - at the bottom (top).

[1] This is just one example. Other data warehouse organizations are also possible for this example.

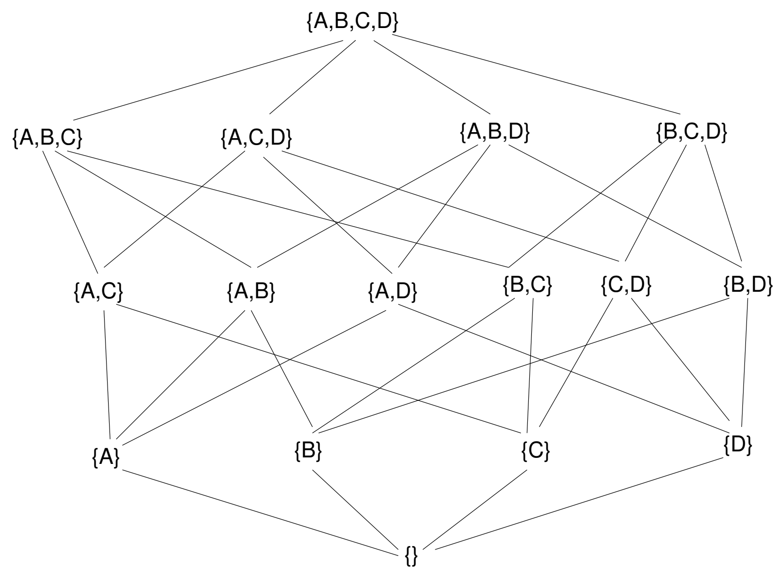

Figure 1: Lattice of cuboids for a four-dimensional data cube.

Example. Consider a four-dimensional datacube $X(A, B, C, D)$. Its lattice of cuboids is shown in Figure 1.

Example. Consider the fact table `RentalCube3D` described above. This table defines a 3D data cube with dimensions *Movie Title*, *Store Location* and *Time*. There are three 2D cuboids in this table:

⟨ *Movie Title, Store Location* ⟩:	Total rentals by movie title and location
⟨ *Movie Title, Time* ⟩:	Renatals of each movie by date
⟨ *Store Location, Time* ⟩:	Overall rentals at each location by date

Stars and Snowflakes

An E-R diagram of a relational database is typically a bipartite graph that connects nodes of type ”entity set” with nodes of type ”relationship set” and with ”relationship set” nodes typically having a degree of two (binary relationships).

In data warehouses, fact tables often contain only foreign keys, rather than full information about individual dimensions. Thus, data warehouses are usually completed by dimension tables, which are associated with the fact table.

The E-R model design for data warehouses often takes the form of either a star or a snowflake.

Star model. The most popular data representation for a data warehouse model is a *star* where the fact table plays the role of the center of the star, connected with ”rays” to the descriptions of individual dimensions (entity sets or perspectives such as time or location). The center relation is normalized and contains no redundant data.

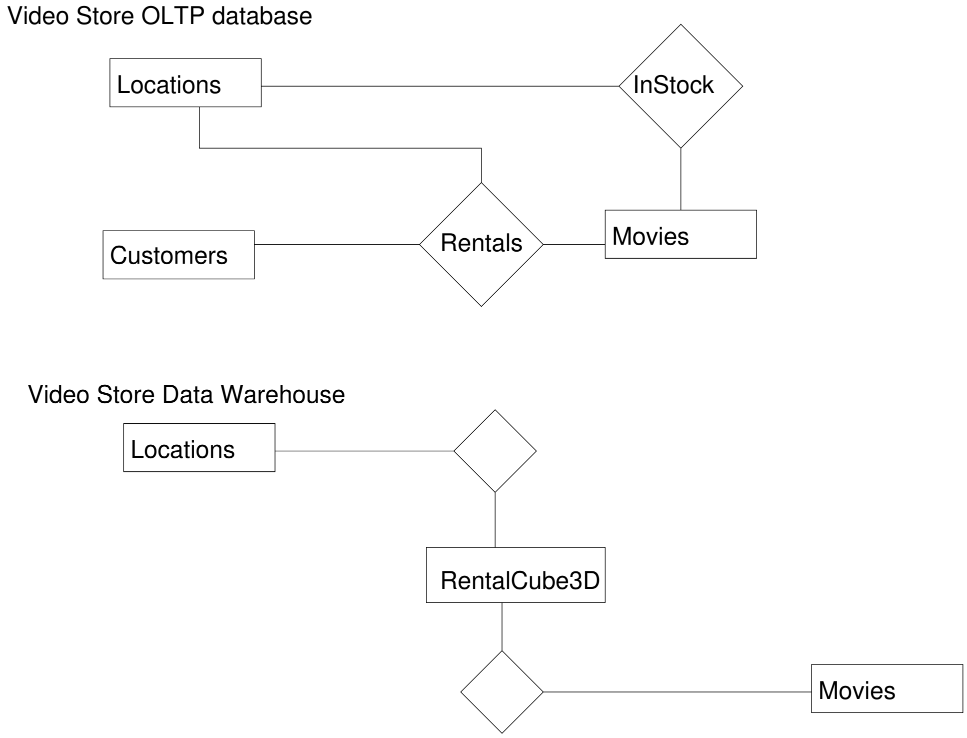

Figure 2: E-R Models for the video rentals store chain.

Snowflake model. Sometimes it is impossible to normalize individual dimension-describing relations in a star model. If we normalize a dimension-describing relations by breaking it into two or more relations, we get a *snowflake model* of data warehouse data.

Example. Let us compare the E-R models of the operational database for the video rentals store and the data warehouse. Figure 2 shows the E-R models for the transactional database (top) and the data warehouse.

Note: Not all dimensions from the data warehouse warrant own *dimension tables*. Some, like *time/date* dimensions exist solely in the fact table.

Concept Hierarchies

Some dimensions may be subject to concept hierarchies: a tree of terms identifying specific values/entities residing in the given dimension.

For example, a concept hierarchy for movie titles in our database, can break all movies by *genre* into `drama`, `comedy`, `action`, `horror`, `family`. Separately, we may have another hierarchy for movie titles, *type*, which breaks movies into `major motion pictures`, `independent`, `foreign` and `B-movies`.

Another dimension, location, can have a more complex hierarchy associated with it: locations are distinguished by *state* at the top level, by *county* within states, by *town* within counties and by *subdivision* within towns.

OLAP operations

Roll-up (**drill-up**): perform aggregation by climbing up a concept hierarchy, or by reducing dimensions.

Example: aggregate video rentals data by genre or aggregate video renatls data by location, adding up the rental numbers for all dates.

```
SELECT Movie, Location, SUM(NumRentals)
FROM RentalsCube3D r
GROUP BY Movie, Location;
```

Drill-down: Reverse roll-up. Introduces a new dimension or a new level of concept hierarchy.

Example: introduce a medium (DVD, VHS, BlueRay) dimension into a 3D title- location - time data cube.

New dimension introduction typically involved combing the original database.

Slice and Dice: Selection (returns a subcube).

Example: return information on rentals of horror movies in California branches by county.

```
SELECT County, SUM(NumRentals)
FROM RentalsCube3D r, Locations l
WHERE r.location = l.Id AND l.State = 'CA'
GROUP BY l.County
```

Pivot (**rotate**): changes orientation of data. The data cube does not change, but the viewpoint from which it is viewed is shifted.

Example: replace the movie title by location by time view with the new time by movie title by location view.

References

[1] W.H. Inmon. *Building the Data Warehouse*, New York, John Wiley & Sons, 1996.