Unit 5

Lecture 35: Data Visualization and Overall Perspective

What is Data Visualisation?

- Data Visualisation is used in software applications to provide an intuitive graphical interface.
- It is applied to many areas to enable users to glean Useful information from their data for faster, more informed decision making
- These areas include: military, private business sector & scientific research

What are the benefits of Data Visualisation?

- Data Visualisation allows users to see several different perspectives of data.
- Data Visualisation makes it possible to interpret vast amount of data
- offers the ability to note exceptions in data
- allows the user to analyse visual patterns in data
- exploring trends with in a database through visualisation by letting analyst navigate through data and visual orient themselves to the patterns in the data.
- can help translate data patterns into insights, making it highly effective decision tool

Aggregation

- Aggregation is the process of consolidating data values into a single value.
- For example, sales data aggregated to the week level, the week data could be aggregated to the month level, and so on.
- Aggregation is a synonym of summarization,
- Data aggregation helps company data warehouses try to piece together different kinds of data within the data warehouse so that they can have meaning that will be useful as statistical basis for company reporting and analysis.

Aggregations can be generated by:

- Developing ETL software, to summarize data from data warehouse source tables and reload the summary tables each time a new data warehouse load is completed;
- Creating views to summarize data from data warehouse source tables each time the view is called.
- Creating materialized views, to summarize from data warehouse source tables each time a new data warehouse load is completed.

Approaches to aggregation:

1. No Aggregation
2. Selective Aggregation
3. Exhaustive Aggregation

1. **No aggregation:** If the volume of data in the fact table will be small enough so that performance is acceptable without aggregates, then no aggregation is applied.
2. **Exhaustive aggregation:** This approach will produce optimal query results because a query can read the minimum number of rows required to return an answer.

Example:

Customer
All Customer
Market
Customer

Sales Geography
Worldwide sales
Country
Region
District
Office
Salesperson

Product
All Product
Product line
Product

Time
Year
Quarter
Month
Week
Day

In the above 4 tables

Total number of possible aggregates are:

= (3 * 6* 3 * 5)

= 270

How to improve Aggregation performance

To improve aggregation performance in the warehouse, the following extensions to the GROUP BY clause is provided, such as

- CUBE and ROLLUP
- GROUPING functions
- GROUPING SETS

- The ROLLUP calculates aggregations such as SUM, COUNT, MAX, MIN, and AVG at increasing levels of aggregation, from the most detailed up to a grand total.
- CUBE is an extension similar to ROLLUP, enabling a single statement to calculate all possible combinations of aggregations.
- GROUPING functions help you identify the group each row belongs to.

Current and Historical Information within Data Warehouse

Challenges for DW managers

- One of the biggest challenges that data warehouse managers face today is the issue of how to manage dimensional tables over a given period of time.
- DW managers also be able to manage this information with current data.
- There are a number of basic modeling techniques that are used, and one example of this is slowly changing dimensions, which can come in type 2 or 3.
- The SCD 2 technique can be utilized to display a dimensional table when a change within similar columns needs to be analyzed over a given period of time.
- The SCD 2 method will use keys within the table that will not change.

Query Facility

The Data Warehouse Query Facility is a tool that allows clients to retrieve data from the Data Warehouse by selecting the fields desired and, in some cases, entering values as a parameter. It is point and click methodology thereby eliminating the need to know a programming language such as SQL (Structured Query Language).

Along with this a data mining query language can be used to specify data mining tasks. In particular, we examine an SQL-based data mining query language called DMQL which contains language primitive for defining data warehouses and data marts.

A DMQL can provide the ability to support ad-hoc and interactive data mining. It has a foundation for system development and evolution. It facilitates information exchange, technology transfer, and commercialization. Data warehouses and data marts can be defined using two language primitives, one for *cube definition* and one for *dimension definition*.

The cube definition statement has the following syntax.

define cube (cube_name> [(dimension_list>]: (measure_list)

The dimension definition statement has the following syntax.

define dimension (dimension_name) as ((attribute_or_subdimension_list])

The star schema of Figure 1 is defined in DMQL as follows.

define cube sales_star (period, item, branch, location): dollars_sold = sum(sales_in_dollars), units_sold = count(*) **define dimension** period 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, state, country)

The define cube statement defines a data cube called sales_star, which correspond to the central sales fact table. This command specifies the dimensions and the two measures dollars_sold and units_sold. The data cube has four dimensions, namely, time, item, branch and location. A define dimension statement is used to define each of the dimensions.

Schemas for Multidimensional Databases

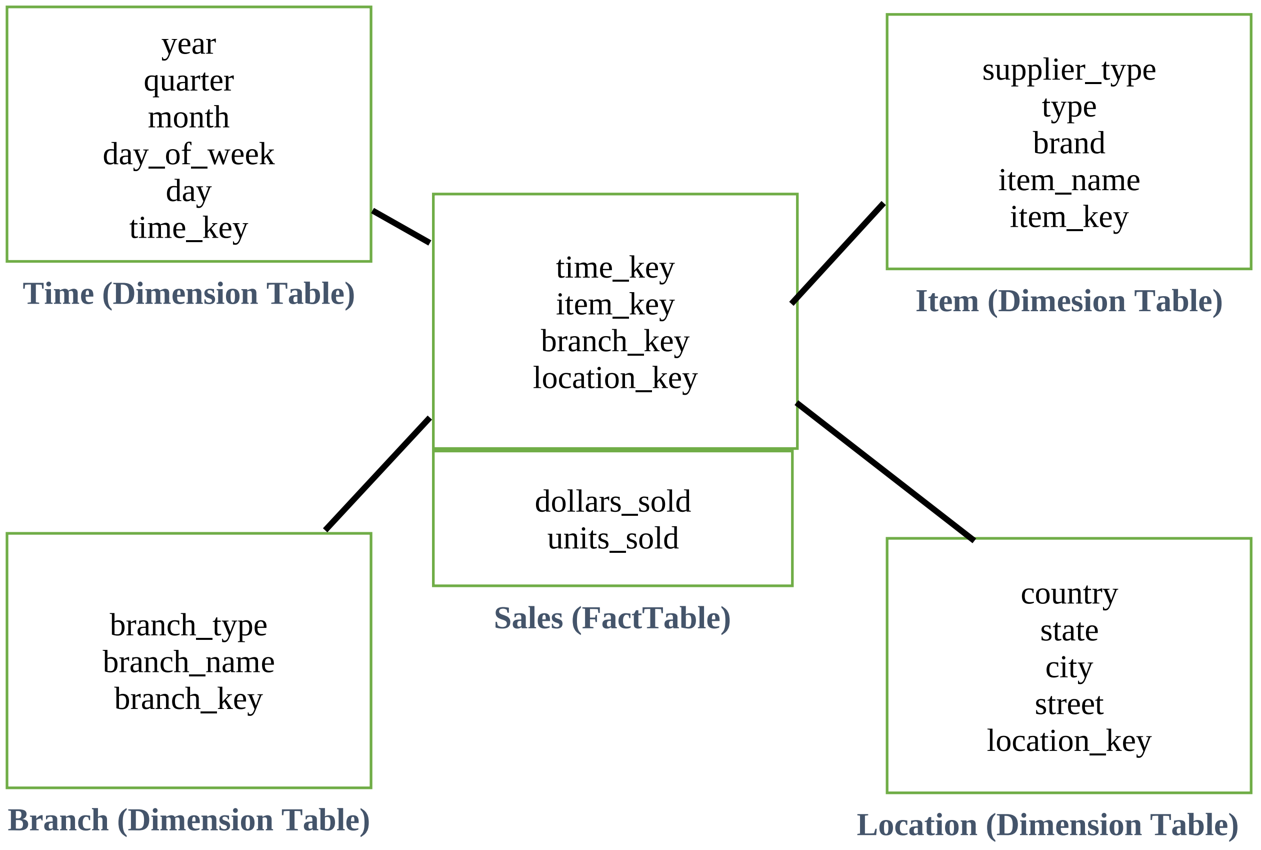

Figure 1 Star Schema for fact Transaction