

The background features a large, faint watermark of the University of Udine logo, which includes a central figure and the text 'UNIVERSITAS STUDIORUM UDINENSIS'.

DMIF, University of Udine

Data Warehousing

Andrea Brunello

andrea.brunello@uniud.it

April, 2020 *(slightly modified by Dario Della Monica)*



- 1 Introduction
- 2 Data Warehouse Fundamental Concepts
- 3 Data Warehouse General Architecture
- 4 Data Warehouse Development Approaches
- 5 The Multidimensional Model
- 6 Operations over Multidimensional Data



Nowadays, most of large and medium size organizations are using information systems to implement their business processes.

As time goes by, these organizations produce a lot of data related to their business, but often these data are not integrated, been stored within one or more platforms.

Thus, they are hardly used for decision-making processes, though they could be a valuable aiding resource.

A central repository is needed; nevertheless, traditional databases are not designed to review, manage and store historical/strategic information, but deal with ever changing operational data, to support “daily transactions”.



What is Data Warehousing?

Data warehousing is a technique for collecting and managing data from different sources to provide meaningful business insights.

It is a blend of components and processes which allows the strategic use of data:

- Electronic storage of a large amount of information which is designed for query and analysis instead of transaction processing
- Process of transforming data into information and making it available to users in a timely manner to make a difference



Why Data Warehousing?

A 3NF-designed database for an inventory system has many tables related to each other through foreign keys.

A report on monthly sales information may include many joined conditions.

This can quickly slow down the response time of the query and report, especially with millions of records involved.

A data warehouse provides a new design which can help to reduce the response time and helps to enhance the performance of queries for reports and analytics.



Who needs a Data Warehouse?

Data warehousing is needed for:

- Decision makers who rely on massive amounts of data
- Users who use customized, complex processes to obtain information from multiple data sources
- Users that need fast performance on a huge amount of data which is a necessity for reports, grids or charts
- Data warehouse is a first step if you want to discover *hidden patterns* in your data (e.g., through data mining)



Business Intelligence (BI) can be defined as a set of tools and techniques for the transformation of raw data into meaningful and useful pieces of information for business analysis purposes.

BI entails the management of large amounts of data to help in identifying, improving, and possibly defining new strategic business opportunities.

In particular, it aims at providing historical and current views of business operations.

It is thus **descriptive**.



BI Related Decisions

Example, considering a customer churn analysis problem:

- How many customers left our company during last year?
- What are our most valuable customers?
- Does our pricing impact the churn rate?





Business analytics (BA) relies on data mining and machine learning to determine what the future will probably look like.

Also, by means of such tools, it may help identify the main reasons behind specific phenomena (e.g., explanation models).

It is thus **predictive**.

Typical questions, churn example:

- How will a 10% increase of the price affect the churn rate?
- Which customers are more likely to leave our company?



Prescriptive analytics is a relatively recent development of Business Analytics.

It goes beyond predicting future outcomes, by also suggesting actions to benefit from the predictions and showing the implications of each decision option.

In the churn example:

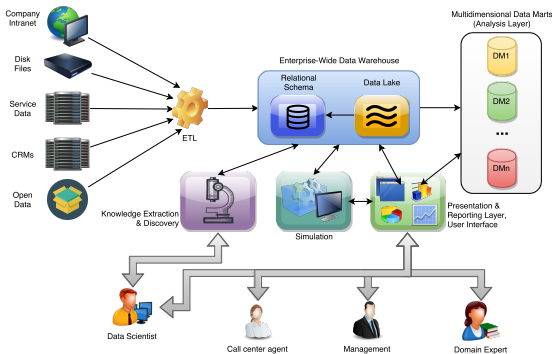
- What are the variables that are most likely to affect the decision of the customer?



Decision Support Systems

A data warehouse is typically the central component of a *Decision Support System*, i.e., an (descriptive / predictive / prescriptive) information system that supports business or organizational decision-making activities.

E.g., providing monitoring tools, graphs, reports, simulations.





From Bookkeeping to Magnetic Tapes

Before electronic data processing, companies used to manage their customers, purchases and inventory using traditional bookkeeping methods .

Early electronic data processing came about in the 1950s, initial systems were based on punch cards, naturally exposed to damage and loss of data.

From 1960s magnetic tapes gave better data storage techniques, but sequential access required full tape scan even for 5% of data. Many dedicated hardware as many formats available.



Direct Access Storage Device

The 1970s saw the advent of disk storage, or the direct access storage device (DASD).

No need to go through records 1, 2, 3, ... n to get to record $n + 1$ once the location address of $n + 1$ is known.

The time to locate a record on a DASD could be measured in milliseconds.

Along with the DASD, it came a new type of system software known as a database management system (DBMS).



With DBMS it was easier to store and access data on a DASD, moreover, the DBMS took care of tasks such as storing data on a DASD, indexing data, and so forth.

By the mid-1970s, applications like bank teller systems and manufacturing control systems became possible (online transaction processing – OLTP).

At the same time, Edgar. F. Codd published his paper on the relational model of systems for managing data, then OLTP systems were oriented to the relational model.



Wal-Mart and the Birth of the DW

- Around 1990 Wal-Mart began to achieve wide acclaim for its mastery of supply chain management.
- Behind this success was Wal-Mart's data warehouse.
- Data is collected by its point-of-sales systems to achieve unprecedented insight into the purchasing habits of its 100 million customers and the logistics guiding its 25,000 suppliers.
- Wal-Mart's data warehouse was the first commercial Enterprise Data Warehouse to reach 1 terabyte of data in 1992.

Data Warehouse Fundamental Concepts



According to William Inmon, a data warehouse is a *subject-oriented, integrated, consistent, non-volatile, and time-variant collection of data* in support of management's decisions.

The analyst job in the data warehouse environment is easier than in the legacy environments:

- single integrated source of data
- granular data is easy (and fast) accessible

The data warehouse is at the heart of the *decision support system* (DSS) operation.



The data warehouse focuses on enterprise-specific *concepts*, as defined in the high-level corporate data model. Subject areas may include:

- Customer
- Product
- Order
- Claim
- Account

Conversely, operational databases hang on enterprise-specific *applications*, meaning that data in them is typically organized by business processes, around the workflows of the company.



Data is fed from multiple, disparate sources into the data warehouse.

As the data is fed, it is converted, reformatted, resequenced, summarized, and so forth (ETL – Extract, Transform, Load).

Data is entered into the data warehouse in such a way that the many inconsistencies at the operational level are resolved.

Consistency applies to all application design issues, such as naming conventions, key structure, measurement of attributes, and physical characteristics of data.



After the data is inserted in the warehouse it is neither changed nor removed.

The only exceptions happen when false data is inserted or the capacity of the data warehouse is exceeded and archiving becomes necessary.

This means that data warehouses can be essentially viewed as read-only databases.

When subsequent changes occur, a new snapshot record is written. In doing so, a historical record of data is kept in the data warehouse.



Time variability implies that the warehouse stores data representative as it existed at many points in time in the past.

A time horizon is the length of time data is represented in an environment; a 5-to-10-year time horizon is normal for a data warehouse.

While operational databases contain current-value data, data warehouses contain sophisticated series of snapshots, each snapshot taken at a specific moment in time.



OLTP queries are typical of operational, daily systems.

This kind of queries generally read or write a small number of tuples, executing transactions on detailed data.

A typical OLTP transaction in a banking environment may be the transfer of money from one account to another.

The four ACID properties (Atomicity, Consistency, Isolation, Durability) are essential for this kind of application, because otherwise money may for example get lost or doubled.

“On-line” means that the analyst should obtain a response in almost real time.



OLAP: On-Line Analytical Processing

On the contrary, the type of query generally executed in data warehouses is OLAP.

In OLAP applications the typical user is not interested in detailed data, but usually in aggregating data over large sets.

E.g., calculate the average amount of money that customers under the age of 20 withdrew from ATMs in a certain region.

OLAP data originates from data found at the operational level, but it is denormalized, summarized, and shaped by the requirements of the management (*multidimensional data*).

This typically requires complex and time consuming transactions to pre-process data.

OLAP queries do not change data warehouse content.



Complex Query

```
SELECT
  CASE WHEN ((SUM(inventory.closed_on_hand) + SUM(changes.received) + SUM(changes.adjustments) +
SUM(changes.transferred_in-changes.transferred_out)) <> 0) THEN ROUND((CAST(SUM(changes.sold_and_closed +
changes.returned_and_closed) AS numeric) * 100) / CAST(SUM(starting.closed_on_hand) + SUM(changes.received) +
SUM(changes.adjustments) + SUM(changes.transferred_in-changes.transferred_out) AS numeric), 5) ELSE 0 END AS "Percent_Sold",
  CASE WHEN (SUM(changes.sold_and_closed) <> 0) THEN ROUND(100*((SUM(changes.closed_markdown_units_sold)*1.0) /
SUM(changes.sold_and_closed)), 5) ELSE 0 END AS "Percent_of_Units_Sold_with_Markdown",
  CASE WHEN (SUM(changes.sold_and_closed * _sku.retail_price) <> 0) THEN
ROUND(100*(SUM(changes.closed_markdown_dollars_sold)*1.0) / SUM(changes.sold_and_closed * _sku.retail_price), 5) ELSE 0 END AS
"Markdown_Percent",
  '0' AS "Percent_of_Total_Sales",
  CASE WHEN SUM((changes.sold_and_closed + changes.returned_and_closed) * _sku.retail_price) IS NULL THEN 0 ELSE
SUM((changes.sold_and_closed + changes.returned_and_closed) * _sku.retail_price) END AS "Net_Sales_at_Retail",
  '0' AS "Percent_of_Ending_Inventory_at_Retail", SUM(inventory.closed_on_hand * _sku.retail_price) AS
"Ending_Inventory_at_Retail",
  "_store"."label" AS "Store",
  "_department"."label" AS "Department",
  "_vendor"."name" AS "Vendor_Name"
FROM
  inventory
  JOIN inventory as starting
    ON inventory.warehouse_id = starting.warehouse_id
    AND inventory.sku_id = starting.sku_id
  LEFT OUTER JOIN
    ( SELECT warehouse_id, sku_id,
      sum(received) as received,
      sum(transferred_in) as transferred_in,
      sum(transferred_out) as transferred_out,
      sum(adjustments) as adjustments,
      sum(sold) as sold
    FROM movement
    WHERE movement.movement_date BETWEEN '2010-08-05' AND '2010-08-19'
    GROUP BY sku_id, warehouse_id ) as changes
    ON inventory.warehouse_id = changes.warehouse_id
    AND inventory.sku_id = changes.sku_id
  JOIN _sku ON _sku.id = inventory.sku_id
  JOIN _warehouse ON _warehouse.id = inventory.warehouse_id
  JOIN _location_hierarchy AS _store ON _store.id = _warehouse.store_id
    AND _store.type = 'Store'
  JOIN _product ON _product.id = _sku.product_id
  JOIN _merchandise_hierarchy AS _department
    ON _department.id = _product.department_id AND _department.type = 'Department'
```




PRIMITIVE DATA (OLTP)

- Application-oriented (transaction-driven)
- Detailed
- Accurate, as of the moment of access
- Can be updated
- Accessed a unit at a time
- Nonredundancy
- Supports day-to-day operations

DERIVED DATA (OLAP)

- Subject-oriented (analysis-driven)
- Summarized
- Represents values over time, snapshots
- Is not meant to be updated
- Accessed a set at a time
- Redundancy is a fact of life
- Supports managerial needs



What is a Data Mart?

A data mart is focused on a single functional area of an organization and contains a subset of data stored in a Data Warehouse.

A data mart is designed for use by a specific department, unit or set of users in an organization. E.g., Marketing, Sales, HR or finance. It is often controlled by a single department in an organization.

Data Mart usually draws data from only a few sources compared to a Data warehouse. Data marts are small in size and are more flexible compared to a Data Warehouse.

Information in a Data Mart is stored according to the multidimensional model (fact and dimension tables).



Data Warehouse vs. Data Mart - 1

Parameter	Data Warehouse	Data Mart
Definition	A Data Warehouse is a large repository of data collected from different organizations or departments within a corporation.	A data mart is an only subtype of a Data Warehouse. It is designed to meet the need of a certain user group.
Usage	It helps to take a strategic decision.	It helps to take tactical decisions for the business.
Objective	The main objective of Data Warehouse is to provide an integrated environment and coherent picture of the business at a point in time.	A data mart mostly used in a business division at the department level.
Designing	The designing process of Data Warehouse is quite difficult. May or may not use in a dimensional model. However, it can feed dimensional models.	The designing process of Data Mart is easy. It is built focused on a dimensional model using a start schema.
Data Handling	Data warehousing includes large area of the corporation which is why it takes a long time to process it.	Data marts are easy to use, design and implement as it can only handle small amounts of data.
Focus	Data warehousing is broadly focused all the departments. It is possible that it can even represent the entire company.	Data Mart is subject-oriented, and it is used at a department level.
Data type	The data stored inside the Data Warehouse are always detailed when compared with data mart.	Data Marts are built for particular user groups. Therefore, data short and limited.
Subject-area	The main objective of Data Warehouse is to provide an integrated environment and coherent picture of the business at a point in time.	Mostly hold only one subject area- for example, Sales figure.



Data Warehouse vs. Data Mart - 2

Parameter	Data Warehouse	Data Mart
Subject-area	The main objective of Data Warehouse is to provide an integrated environment and coherent picture of the business at a point in time.	Mostly hold only one subject area- for example, Sales figure.
Data storing	Designed to store enterprise-wide decision data, not just marketing data.	Dimensional modeling and star schema design employed for optimizing the performance of access layer.
Data type	Time variance and non-volatile design are strictly enforced.	Mostly includes consolidation data structures to meet subject area's query and reporting needs.
Data value	Read-Only from the end-users standpoint.	Transaction data regardless of grain fed directly from the Data Warehouse.
Scope	Data warehousing is more helpful as it can bring information from any department.	Data mart contains data, of a specific department of a company. There are maybe separate data marts for sales, finance, marketing, etc. Has limited usage
Source	In Data Warehouse Data comes from many sources.	In Data Mart data comes from very few sources.
Size	The size of the Data Warehouse may range from 100 GB to 1 TB+.	The Size of Data Mart is less than 100 GB.
Implementation time	The implementation process of Data Warehouse can be extended from months to years.	The implementation process of Data Mart is restricted to few months.



What is a Data Lake?

A Data Lake is a storage repository that can store large amounts of structured, semi-structured, and unstructured data.

- It is a place where to store every type of data in its native format with no fixed limits on size or type
- It allows to access data before the ETL process, thus it retains all data coming from the sources
- Data is only transformed when the user is about to use it (*schema on read*, vs. *schema on write* in the data warehouse)
- Storing information in a data lake is relatively inexpensive with respect to storing them in a data warehouse



A Data Lake is not a substitute for a Data Warehouse.

For instance, a Data Warehouse guarantees quick answers for interactive queries thanks to the schema on write approach.

A Data Lake can grow without control and become useless (data swamp)... store only the possibly useful information!



Data Lake vs. Data Warehouse

Parameters	Data Lake	Data Warehouse
Storage	In the data lake, all data is kept irrespective of the source and its structure. Data is kept in its raw form. It is only transformed when it is ready to be used.	A data warehouse will consist of data that is extracted from transactional systems or data which consists of quantitative metrics with their attributes. The data is cleaned and transformed
History	Big data technologies used in data lakes is relatively new.	Data warehouse concept, unlike big data, had been used for decades.
Data Capturing	Captures all kinds of data and structures, semi-structured and unstructured in their original form from source systems.	Captures structured information and organizes them in schemas as defined for data warehouse purposes
Data Timeline	Data lakes can retain all data. This includes not only the data that is in use but also data that it might use in the future. Also, data is kept for all time, to go back in time and do an analysis.	In the data warehouse development process, significant time is spent on analyzing various data sources.
Users	Data lake is ideal for the users who indulge in deep analysis. Such users include data scientists who need advanced analytical tools with capabilities such as predictive modeling and statistical analysis.	The data warehouse is ideal for operational users because of being well structured, easy to use and understand.
Storage Costs	Data storing in big data technologies are relatively inexpensive then storing data in a data warehouse.	Storing data in Data warehouse is costlier and time-consuming.
Task	Data lakes can contain all data and data types; it empowers users to access data prior the process of transformed, cleansed and structured.	Data warehouses can provide insights into pre-defined questions for pre-defined data types.
Processing time	Data lakes empower users to access data before it has been transformed, cleansed and structured. Thus, it allows users to get to their result more quickly compares to the traditional data warehouse.	Data warehouses offer insights into pre-defined questions for pre-defined data types. So, any changes to the data warehouse needed more time.
Position of Schema	Typically, the schema is defined after data is stored. This offers high agility and ease of data capture but requires work at the end of the process	Typically schema is defined before data is stored. Requires work at the start of the process, but offers performance, security, and integration.

Data Warehouse General Architecture

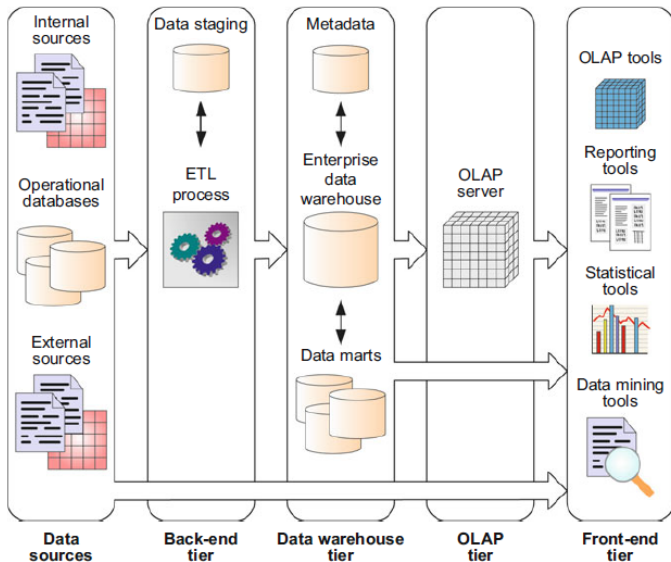


A modern general data warehouse architecture typically consists of several tiers:

- *The back-end tier* includes extraction, transformation, and loading (ETL) tools and a data staging area
- *The data warehouse tier* is composed of an enterprise data warehouse and/or several data marts and a metadata repository
- *The OLAP tier* is composed of an OLAP server, which provides a multidimensional view of the data
- *The front-end tier* is used for data analysis and visualization. It contains client tools such as OLAP tools, reporting tools, statistical tools, and data mining tools



Data Warehouse Architecture Schema





In the back-end tier, the process known as extraction, transformation, and loading (ETL) is performed:

- *extract*: data is gathered from multiple, heterogeneous sources
- *transform*: data cleansing (errors and inconsistencies removal), integration (data reconciliation, e.g., formats) and aggregation (to the level of granularity of the data warehouse)
- *load*: regularly feed the data warehouse the new data

Such operations are typically performed within a *data staging area*, i.e., an intermediate database.



- Data from different sources may be inconsistent:
 - "Chris Date", "C Date", "C. J. Date" ...
 - "1 inch", "2.54 cm", ...
- Even data from the same database may be inconsistent
- Techniques like Fuzzy lookup look for approximate matches
- Merging data from different sources may introduce duplicates
- Merging data from different sources may clash, as because of progressive IDs



Before processing all the dirty data, it is important to determine the cleansing cost for every dirty data element.

Every organization would like to have all the data clean, but it's also a matter of time and cost. If cleaning it all would simply take too long, it is better not to try to cleanse all the data.

It is important not to skip the cleansing process altogether because the biggest reason for building the Data Warehouse is to offer cleaner and more reliable data.

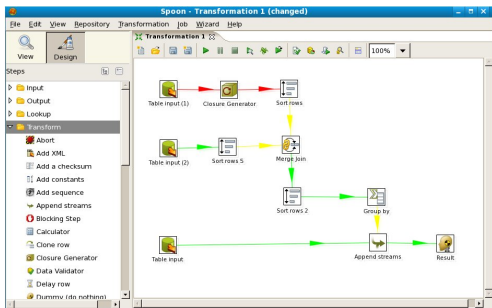
It may worth to use auxiliary tables and indexes to speed up all the operations involved in ETL phase (data staging area design is also important).



QuerySurge is built specifically to automate the testing of Data Warehouses & Big Data.

MarkLogic is a NoSQL data warehousing solution that includes a fully-fledged data integration and data management solution.

Pentaho Data Integration / Talend Open Studio support the creation of complex ETL workflows.





Components:

- *The enterprise data warehouse (EDW)* is centralized and encompasses the entire organization
- *The data marts* are smaller, specialized data warehouses targeted toward a particular functional or departmental area in the organization (data can be derived from the EDW or collected directly from data sources)
- *The metadata repository* contains information about the data warehouse, data sources and ETL processes (schema definitions, monitoring information, security information, data lineage, ...).



The OLAP tier is composed of an OLAP server, which presents users and applications with multidimensional data from data warehouses or data marts.

- data is (multi)dimensional, regardless of its format in the data warehouse tier
- the tier allows navigation/analysis/reporting from data
- there is not yet a standardized language for defining and manipulating data cubes
- XMLA (XML for Analysis) aims at providing a common language for exchanging multidimensional data between client applications and OLAP servers
- MDX (MultiDimensional eXpressions) is a query language for OLAP databases



The front-end tier contains client tools that allow users to exploit the contents of the data warehouse:

- *OLAP tools* allow interactive exploration and manipulation of the warehouse data
- *Reporting tools* enable the production, delivery, and management of (interactive) reports (using predefined queries), KPI indicators, etc.
- *Statistical tools* are used to analyze and visualize the cube data using statistical methods
- *Data mining tools* allow users to analyze data in order to discover valuable knowledge and to make predictions

Data Warehouse Development Approaches

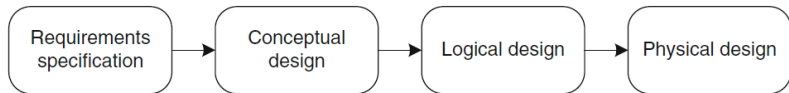


There are two major approaches to the design of data warehouses and data marts:

- *top-down (Inmon)*: first, the DW is developed through the use of relational modelling techniques. Then, data-marts are created, that extract subsets of information from the DW, filtering and aggregating it as necessary.
- *bottom-up (Kimball)*: the (multi)dimensional data marts that serve the analytical needs of enterprise departments are first built. The DW results as the conglomerate of all data marts within the enterprise. Their integration is achieved through the *Information Bus* (= all data marts must use standardized dimensions).



Phases in Data Warehouse Design



The Multidimensional Model



The multidimensional Model

The distinctive features of OLAP applications suggest the adoption of a multidimensional representation of data, since running analytical queries against traditionally stored information would result in complex query specification and long response times.

The multidimensional model relies on the concepts of *fact*, *measure*, and *dimension*, and makes use of two kinds of tables: *fact tables* and *dimension tables*.



In a data warehouse context, a *fact* is the part of your data that indicates a specific event or transaction that has happened, like the sale of a product, or receiving a shipment.

A fact is composed of multiple numerical *measures*, that describe it.

As an example, a fact may be receiving an order for some shoes, detailed by the measures 'price' and 'quantity'.



A *dimension* is a structure that categorizes/indexes facts in order to enable users to answer business questions.

For instance, common dimensions may refer to: time, place, product.

The primary functions of dimensions are to provide, e.g., filtering and grouping.

Typically dimensions are organized internally into one or more hierarchies. For instance, *date* may have the hierarchy:

- Days (grouped into) Months (grouped into) Years



Going back to our previous example, the order may be detailed by the following 2 *measures*, and 3 *dimension attributes*:

- total amount US\$ 750
- quantity purchased is 10
- received yesterday at 2 pm
- served by our store in New York
- placed by customer #XAZ19



Fact Table:

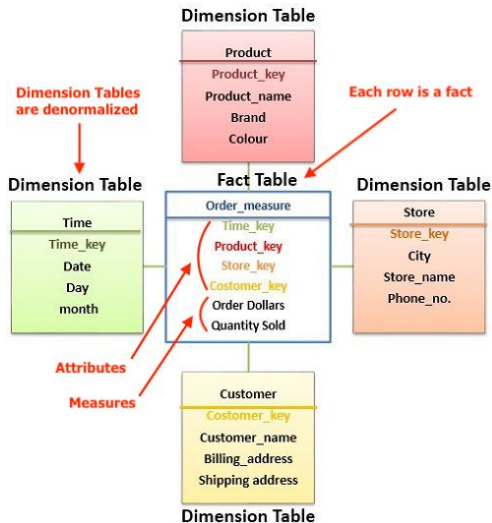
- A fact table is a central table in a dimensional model
- It contains facts/measures and foreign keys to dimension tables

Dimension Table:

- A dimension table contains the dimensions of a fact
- Dimension tables are denormalized
- There is no limit on the number of dimensions
- The dimension can also contain one or more hierarchical relationships



Example (Star Schema)





Multidimensional schemas are specifically designed to address the unique needs of very large databases designed for analytical purpose (OLAP).

Main types of multidimensional schemas:

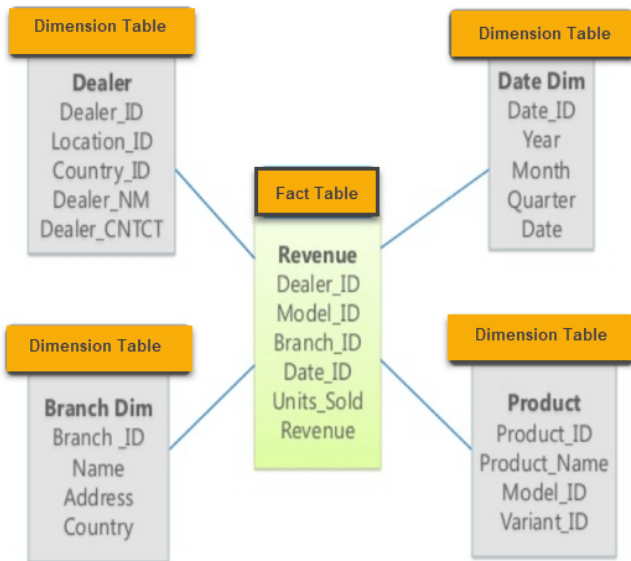
- Star Schema
- Snowflake Schema
- Constellation Schema



- Every dimension in a star schema is represented with a single dimension table
- The dimension table is joined to the fact table using a foreign key
- The dimension tables are not joined to each other
- Fact tables would contain measures and keys pointing to dimensions
- The dimension tables are *not normalized*, which is bad due to data redundancy but efficient for only-read queries



Star Schema Example





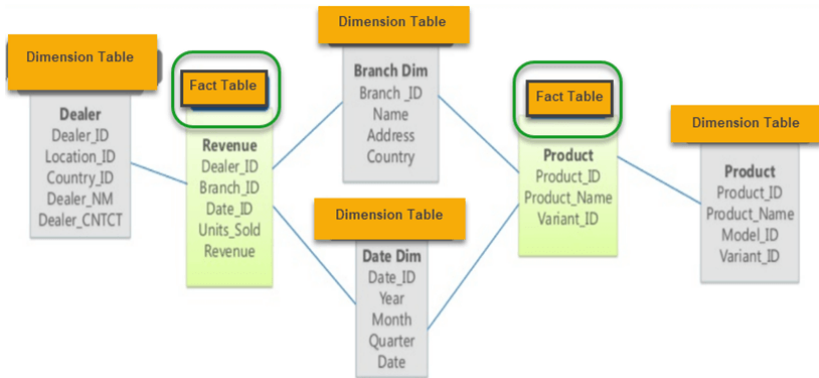
- The difference w.r.t. star schemas is that the dimension tables can be *normalized*, which splits data into additional tables
- The main benefits of the snowflake schema are that there is less redundancy and it uses less disk space
- The primary disadvantage of the snowflake schema is that the additional levels of attribute normalization adds complexity to source query joins, when compared to the star schema



- A constellation schema is a collection of multiple tables sharing dimension tables
- It can be viewed as a collection of star and/or snowflake schemas



Constellation Schema Example



Operations over Multidimensional Data



The Multidimensional Model

In the multidimensional model, data is represented in an n -dimensional space, usually called a data cube or a hypercube.

A data cube is defined by dimensions (cube edges) and facts (cube cells):

- Dimensions are perspectives used to analyze the data (their hierarchies represent the granularity/level of detail)
- Facts have usually related numeric values (the measures)

Data cubes can be sparse: there may not be a cell value for each combination of dimensions.



Multidimensional Data in a Spreadsheet

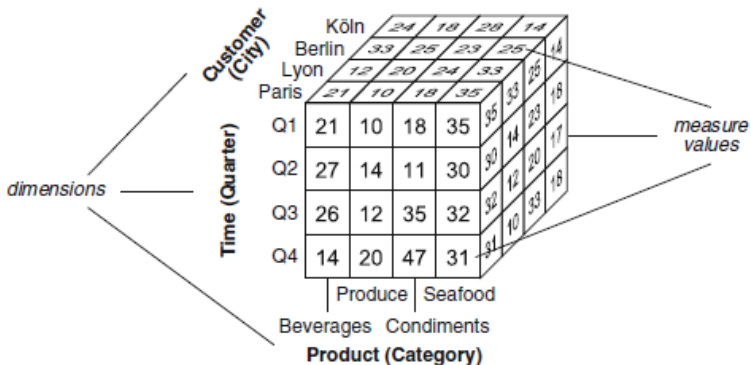
Bi-dimensional pivot table, considering:

- measure 'Amount'
- dimensions 'Place' and 'Product'
- facts are the amount of products sold in each country

	A	B	C	D	E	F	G	H	I	J
1	Category	(All) ▾								
2										
3	Sum of Amount	Column ▾								
4	Row Labels ▾	Apple	Banana	Beans	Broccoli	Carrots	Mango	Orange	Grand Total	
5	Australia	20634	52721	14433	17953	8106	9186	8680	131713	
6	Canada	24867	33775		12407		3767	19929	94745	
7	France	80193	36094	680	5341	9104	7388	2256	141056	
8	Germany	9082	39686	29905	37197	21636	8775	8887	155168	
9	New Zealand	10332	40050		4390			12010	66782	
10	United Kingdom	17534	42908	5100	38436	41815	5600	21744	173137	
11	United States	28615	95061	7163	26715	56284	22363	30932	267133	
12	Grand Total	191257	340295	57281	142439	136945	57079	104438	1029734	
13										



OLAP Cube Example





To extract strategic knowledge from a cube, it is necessary to view its data at several levels of detail.

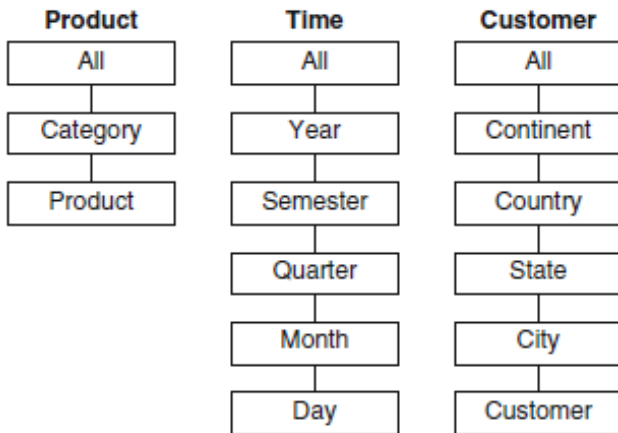
Dimension hierarchies allow to define a sequence of mappings relating lower-level, detailed concepts to higher-level, more general concepts.

As an example, dates of purchase could be aggregated into coarser grained levels of detail, such as months, or years.

Given a hierarchy level, its lower level is called the *child* and its higher level is called the *parent*.

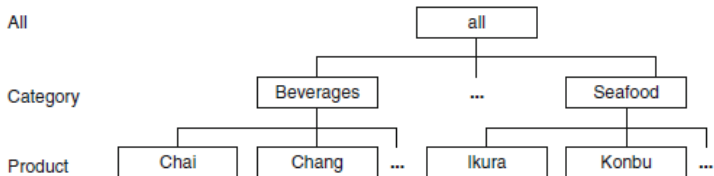


Hierarchy Example - 1





Hierarchy Example - 2





Measures in a cube are associated with *aggregation functions* that combine several values of a measure into a single one.

Aggregation of measures are useful when one changes the level of detail at which data in a cube are visualized (*from finer/lower to coarser/higher level*).

This is performed by traversing the hierarchies of the dimensions.

E.g., measure *total amount* of sales is associated to aggregation function *sum*: to show total amount of a quarter can be obtained by summing total amounts of months.



It refers to the correct aggregation of cube measures along dimension hierarchies, in order to obtain consistent aggregation results. A set of conditions must hold:

- *Disjointness of instances*: the grouping of instances in a hierarchy level must result in disjoint subsets (e.g., a city cannot belong to two provinces)
- *Completeness*: all instances must be included in some group at every hierarchy level and each instance must be related to a parent in the upper level (no broken hierarchies)
- *Correctness*: it refers to the correct use of the aggregation functions (e.g., mind the difference between *sum* and *count*)



Types of Measures - 1

According to the way in which they can be aggregated using addition, measures can be:

- *Additive measures* can be meaningfully summarized along **all the dimensions**, using addition (the most common type, e.g., the amount items that have been sold)
- *Semiadditive measures* can be meaningfully summarized using addition along **some, but not all, dimensions** (inventory quantities can be meaningfully aggregated along spatial dimension but cannot be meaningfully aggregated in the temporal dimension)
- *Nonadditive measures* **cannot be meaningfully summarized** using addition across any dimension (item price, cost per unit, and exchange rate)

Other aggregation measures may work, such as: average, max, min, ... think carefully carefully about this!



Types of Measures - 2

Some caution has also to be taken with the aggregation of intermediate results.

According to incremental aggregation computation, measures can be:

- *Distributive measures*: aggregation function can be computed in a distributed way, i.e., only consider the **immediate lower level values** (count, sum, max, min)
- *Algebraic measures*: aggregation function that can be expressed as a **scalar function of distributive ones** (an example is the average, that can use sum and count/average)
- *Holistic measures*: aggregation function cannot be computed from other subaggregates, i.e., **finest granularity values must be used** (median, mode, rank)



The four types of analytical operations performed on OLAP cubes are:

- Roll-up
- Drill-down
- Pivot (rotate)
- Slice and dice



Roll-up

It involves summarizing the data along a chosen dimension, navigating from a finer level of detail (down) to a coarser one (up).

Diagram (a) shows a 3D cube representing data at the city level. The vertical axis is 'Time (Quarter)' with values Q1, Q2, Q3, and Q4. The horizontal axis is 'Customer (City)' with values Köln, Berlin, Lyon, and Paris. The depth axis is 'Product (Category)' with values Produce, Seafood, Beverages, and Condiments. The data values are as follows:

Time (Quarter)	Customer (City)	Produce	Seafood	Beverages	Condiments
Q1	Köln	21	10	18	35
Q1	Berlin	33	25	23	25
Q1	Lyon	12	20	24	33
Q1	Paris	21	10	18	35
Q2	Köln	27	14	11	30
Q2	Berlin	30	14	23	17
Q2	Lyon	30	12	20	17
Q2	Paris	30	12	20	17
Q3	Köln	26	12	35	32
Q3	Berlin	32	10	33	18
Q3	Lyon	32	10	33	18
Q3	Paris	32	10	33	18
Q4	Köln	14	20	47	31
Q4	Berlin	31	10	33	18
Q4	Lyon	31	10	33	18
Q4	Paris	31	10	33	18

(a) Original

Diagram (b) shows a 3D cube representing data at the country level. The vertical axis is 'Time (Quarter)' with values Q1, Q2, Q3, and Q4. The horizontal axis is 'Customer (Country)' with values Germany and France. The depth axis is 'Product (Category)' with values Produce, Seafood, Beverages, and Condiments. The data values are as follows:

Time (Quarter)	Customer (Country)	Produce	Seafood	Beverages	Condiments
Q1	Germany	33	30	42	68
Q1	France	57	43	51	39
Q2	Germany	39	26	41	44
Q2	France	68	39	41	44
Q3	Germany	30	22	46	44
Q3	France	44	41	37	51
Q4	Germany	25	29	49	41
Q4	France	41	51	41	51

(b) Roll-up to the Country level



Drill-down

It allows the user to navigate among levels of data, ranging from the most summarized (up) to the most detailed (down), along a given hierarchy.

Looking at the detail beneath a summary number may be useful, especially where the summary number is surprising.

		Köln				Berlin				Lyon				Paris			
		24	18	28	14	33	25	23	25	12	20	24	33	25	25	14	14
Time (Quarter)	Q1	21	10	18	35	35	14	23	18	21	10	18	35	35	14	23	18
	Q2	27	14	11	30	30	14	20	17	27	14	11	30	30	14	20	17
	Q3	26	12	35	32	32	12	20	18	26	12	35	32	32	12	20	18
	Q4	14	20	47	31	31	10	33	18	14	20	47	31	31	10	33	18
		Produce				Seafood				Beverages				Condiments			
		Product (Category)															

(c) Original

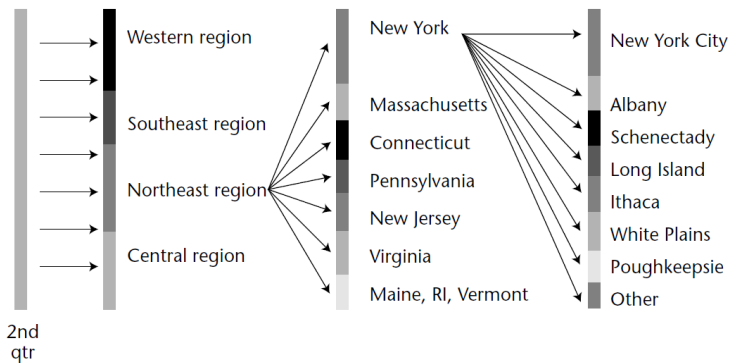
		Köln				Berlin				Lyon				Paris			
		8	6	9	5	10	8	11	8	4	7	8	14	8	14	8	6
Time (Quarter)	Jan	7	2	6	20	20	14	10	7	7	2	6	20	20	14	10	7
	Feb	8	4	8	8	8	10	7	3	8	4	8	8	8	10	7	3
	Mar	6	4	4	7	7	9	6	4	4	7	7	9

	Dec	4	4	16	7	7	5	14	8	4	4	16	7	7	5	14	8
		Produce				Seafood				Beverages				Condiments			
		Product (Category)															

(d) Drill-down to the Month level



Drill-down - Example





This operation allows an analyst to rotate the cube in space to see its various faces.

		Köln				14			
		24	18	28	14				
Customer (City)	Berlin	33	25	23	25	14			
	Lyon	12	20	24	33	25			
	Paris	21	10	18	35	18			
	Q1	21	10	18	35	35	33	25	18
Time (Quarter)	Q2	27	14	11	30	30	14	23	17
	Q3	26	12	35	32	32	12	20	17
	Q4	14	20	47	31	31	10	33	18
			Produce		Seafood		Beverages		Condiments
		Product (Category)							

(e) Original

		Seafood				31			
		35	30	32	31				
Product (Category)	Condiments	18	11	35	47	31			
	Produce	10	14	12	20	47			
	Beverages	21	27	26	14	20	10		
	Q1	21	27	26	14	14	20	47	31
Customer (City)	Paris	21	27	26	14	13	17	21	33
	Lyon	12	14	11	13	13	28	20	18
	Berlin	33	28	35	32	32	19	47	18
	Köln	24	23	25	18	18	19	47	18
		Q1	Q2	Q3	Q4	Time (Quarter)			

(f) Pivot to show Customer vs. Time



It is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, obtaining a new cube with one fewer dimension.

		Köln							
		24	18	28	14				
Customer (City)	Berlin	33	25	23	25	14			
	Lyon	12	20	24	33	25			
	Paris	21	10	18	35	18			
	Q1	21	10	18	35	35	33	23	25
	Q2	27	14	11	30	30	14	23	17
Q3	26	12	35	32	32	12	20	17	
Q4	14	20	47	31	31	10	33	18	
		Produce		Seafood					
		Beverages		Condiments					
		Product (Category)							

(g) Original

Q1	21	10	18	35
Q2	27	14	11	30
Q3	26	12	35	32
Q4	14	20	47	31
	Produce		Seafood	
	Beverages		Condiments	
	Product (Category)			

(h) Slice on City='Paris'



The dice operation is a generalization of slice. More than one dimension can be tested, with multiple conditions.

Time (Quarter)	Customer (City)				Product (Category)			
	Köln	Berlin	Lyon	Paris	Produce	Beverages	Condiments	Seafood
Q1	21	10	18	35	35	23	17	14
Q2	27	14	11	30	30	14	20	17
Q3	26	12	35	32	32	12	33	18
Q4	14	20	47	31	31	10	33	18

(i) Original

Time (Quarter)	Customer (City)				Product (Category)			
	Lyon	Paris	Produce	Seafood	Beverages	Condiments	Seafood	Produce
Q1	21	10	18	35	35	23	17	14
Q2	27	14	11	30	30	14	20	17

(j) Dice on City='Paris' or 'Lyon'
and Quarter='Q1' or 'Q2'



- Extensive usage of materialized views
- Precomputing all the possible aggregations is space and time expensive
- 2^n GROUP BY combination for n dimensions
- It is better to precompute only some aggregated functions and derive the others exploiting the previous ones
- For instance sums wrt (item_name; color) can be obtained from (item_name; color; size)
- However, this is not possible in some cases (e.g., median)



Relational OLAP (ROLAP): data are stored in relational tables. Preaggregation is critical. Operations are mapped to SQL statements. Support from the DBMS (concurrency, scalability, data recovery, ...).

Multidimensional OLAP (MOLAP): multidimensional arrays are used to represent the cubes. Fast response times. Poor storage utilization for sparse data. Less storage capacity than ROLAP.

Hybrid OLAP (HOLAP): systems that keep part of the data in MOLAP (recent data, or preaggregates) and part in ROLAP (old, or detailed data).



- Talend, open source, www.talend.com (ETL)
- Kettle Pentaho, o.s., kettle.pentaho.com (ETL)
- Pentaho Business Analytics, www.pentaho.com
- Weka, o.s., www.cs.waikato.ac.nz/ml/weka (Data mining)
- RapidMiner, o.s., www.rapidminer.com (ETL, data mining)
- Mondrian, o.s., mondrian.pentaho.com (OLAP)
- Palo, o.s., sourceforge.net/projects/palo (MOLAP)
- jPivot, o.s., jpivot.sourceforge.net (client per Mondrian)
- Jasper, o.s., jasperforge.org (ETL, OLAP, . . .)
- Wabit, o.s., code.google.com/p/wabit (OLAP)



- Teradata, www.teradata.com (row/column store)
- Greenplum, www.greenplum.com (row/column store)
- HP Vertica, vertica.com (column store)
- Aster Data, www.asterdata.com (DBMS, map/reduce, R)
- Oracle BI, Essbase, . . . , oracle.com (vari prodotti)
- Sybase IQ, www.sybase.com/products (column store)
- IBM DB2, www.ibm.com/software/data/db2
- IBM Netezza, www.netezza.com
- IBM Cognos, www.ibm.com/software/analytics/
- HP Vertica, vertica.com (column store)
- LucidDB, o.s., www.luciddb.org (column store)
- MonetDB, o.s., www.monetdb.org (column store)
- SciDB, o.s., www.scidb.org



A. Vaisman, E. Zimányi *Data Warehouse Systems - Design and Implementation*, Springer 2014

A. Silberschatz, H. F. Korth, S. Sudarshan *Database system concepts*, §20.1–20.2, §5.5–5.6 6th edition, 2010

W. I. Immon *Building the Data Warehouse Fourth Edition*, Wiley Publishing, Inc. 2005