Data Warehouse Technology

Dr. Abu Raihan Mostofa Kamal

April 23, 2024

Contents

Data warehouse: Definition

According to William H. Inmo

A data warehouse is a **subject-oriented, integrated, time-variant and nonvolatile** collection of data in support of management's decision making process.

Key features from this definition

- **Subject-oriented:** Data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process. Example: only sales per month is important, ignore finer transaction details.
- **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- **Time-variant:** Data are stored to provide information from an historic perspective (e.g., the past 10 years).
- **Nonvolatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. **Due to this separation**, a data warehouse **does not require** transaction processing, recovery, and concurrency control mechanisms. It usually **requires only two operations** in data accessing: i. initial loading of data and ii. access of data.

Key features from this definition

- **Subject-oriented:** Data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process. Example: only sales per month is important, ignore finer transaction details.
- **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- **Time-variant:** Data are stored to provide information from an historic perspective (e.g., the past 10 years).
- **Nonvolatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. **Due to this separation**, a data warehouse **does not require** transaction processing, recovery, and concurrency control mechanisms. It usually **requires only two operations** in data accessing: i. initial loading of data and ii. access of data.

Key features from this definition

- **Subject-oriented:** Data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process. Example: only sales per month is important, ignore finer transaction details.
- **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- **Time-variant:** Data are stored to provide information from an historic perspective (e.g., the past 10 years).
- **Nonvolatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. **Due to this separation**, a data warehouse **does not require** transaction processing, recovery, and concurrency control mechanisms. It usually **requires only two operations** in data accessing: i. initial loading of data and ii. access of data.

Key features from this definition

- **Subject-oriented:** Data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process. Example: only sales per month is important, ignore finer transaction details.
- **Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- **Time-variant:** Data are stored to provide information from an historic perspective (e.g., the past 10 years).
- **Nonvolatile:** A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. **Due to this separation**, a data warehouse **does not require** transaction processing, recovery, and concurrency control mechanisms. It usually **requires only two operations** in data accessing: i. initial loading of data and ii. access of data.

Operational DB & Data Warehouses: Difference

There are basically two approaches in accessing database information:

OLTP : Main design architecture for Operational Database

The major task of online operational database systems are based on online transaction processing (OLTP) systems.They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

OLAP: Main design architecture for Data Warehouse

Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as online analytical processing (OLAP) systems.

Operational DB & Data Warehouses: Difference

There are basically two approaches in accessing database information:

OLTP : Main design architecture for Operational Database

The major task of online operational database systems are based on online transaction processing (OLTP) systems.They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

OLAP: Main design architecture for Data Warehouse

Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as online analytical processing (OLAP) systems.

Operational DB & Data Warehouses: Difference (Cont.1)

- **Users and system orientation:** OLTP is customer-oriented and mainly used by clerks, clients, and information technology professionals. Whereas OLAP is market-oriented and is used for data analysis used by knowledge workers, including managers, executives, and analysts. (General Users Vs. Decision Makers)
- **Data contents:** OLTP contains **too details** of the processing. OLAP generally **summarizes and aggregates** data at different levels of granularity. (Details Vs. Summarized)

Operational DB & Data Warehouses: Difference (Cont.1)

- **Users and system orientation:** OLTP is customer-oriented and mainly used by clerks, clients, and information technology professionals. Whereas OLAP is market-oriented and is used for data analysis used by knowledge workers, including managers, executives, and analysts. (General Users Vs. Decision Makers)
- **Data contents:** OLTP contains **too details** of the processing. OLAP generally **summarizes and aggregates** data at different levels of granularity. (Details Vs. Summarized)

Operational DB & Data Warehouses: Difference (Cont.2)

- **Database design:** OLTP adopts an entity-relationship (ER) data model and an application-oriented database design. OLAP adopts subject-oriented database design (will be discussed later in this chapter).
- **View:** OLTP focuses on the **current data** (within origination), without referring to historic data. OLAP works on **historic data**. It also deals with information that originates from different organizations.
- **Access patterns:** OLTP requires concurrency control and recovery mechanism. OLAP **involves read-only operations** (as it mainly works on historic data).

Operational DB & Data Warehouses: Difference (Cont.2)

- **Database design:** OLTP adopts an entity-relationship (ER) data model and an application-oriented database design. OLAP adopts subject-oriented database design (will be discussed later in this chapter).
- **View:** OLTP focuses on the **current data** (within origination), without referring to historic data. OLAP works on **historic data**. It also deals with information that originates from different organizations.
- **Access patterns:** OLTP requires concurrency control and recovery mechanism. OLAP **involves read-only operations** (as it mainly works on historic data).

Operational DB & Data Warehouses: Difference (Cont.2)

- **Database design:** OLTP adopts an entity-relationship (ER) data model and an application-oriented database design. OLAP adopts subject-oriented database design (will be discussed later in this chapter).
- **View:** OLTP focuses on the **current data** (within origination), without referring to historic data. OLAP works on **historic data**. It also deals with information that originates from different organizations.
- **Access patterns:** OLTP requires concurrency control and recovery mechanism. OLAP **involves read-only operations** (as it mainly works on historic data).

Why A Separate Data Warehouse?

- **High performance of both systems:** Operational database is designed and optimized by the application needs. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.
- **Two modes of operations:** OLTP requires concurrency control and recovery mechanism. OLAP involves read-only operations. OLAP operation in existing OLTP will grade the performance of the regular transactions.
- **Historic data support:** OLTP normally avoids historic data support from different sources. OLAP works on such historic data.

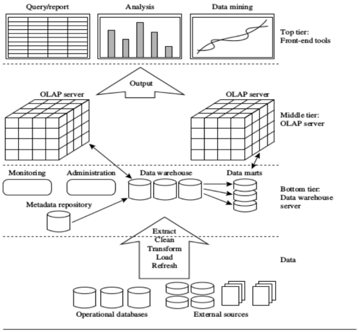
Why A Separate Data Warehouse?

- **High performance of both systems:** Operational database is designed and optimized by the application needs. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.
- **Two modes of operations:** OLTP requires concurrency control and recovery mechanism. OLAP involves read-only operations. OLAP operation in existing OLTP will grade the performance of the regular transactions.
- **Historic data support:** OLTP normally avoids historic data support from different sources. OLAP works on such historic data.

Why A Separate Data Warehouse?

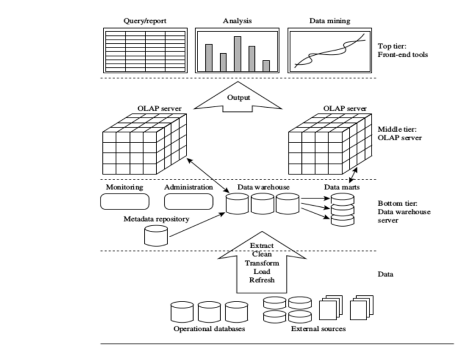
- **High performance of both systems:** Operational database is designed and optimized by the application needs. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.
- **Two modes of operations:** OLTP requires concurrency control and recovery mechanism. OLAP involves read-only operations. OLAP operation in existing OLTP will grade the performance of the regular transactions.
- **Historic data support:** OLTP normally avoids historic data support from different sources. OLAP works on such historic data.

Data Warehousing: A Multitiered Architecture



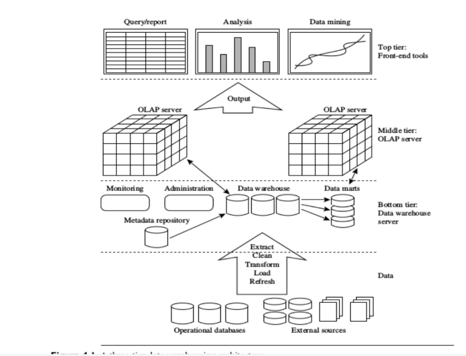
- The bottom tier is almost a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases.Data feeding interface called gateways.
  Common gateways are: jdbc, odbc.
- The middle tier is an **OLAP server** that is typically implemented using either (1) a relational OLAP (ROLAP) or (2) a multidimensional OLAP (MOLAP) model.
- The top tier contains query and **reporting tools,analysis tools**, and/or data mining tools (e.g., trend analysis & prediction).

Data Warehousing: A Multitiered Architecture



- The bottom tier is almost a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases.Data feeding interface called gateways.
  Common gateways are: jdbc, odbc.
- The middle tier is an **OLAP server** that is typically implemented using either (1) a relational OLAP (ROLAP) or (2) a multidimensional OLAP (MOLAP) model.
- The top tier contains query and **reporting tools,analysis tools**, and/or data mining tools (e.g., trend analysis & prediction).

Data Warehousing: A Multitiered Architecture



- The bottom tier is almost a relational database system. Back-end tools and utilities are used to feed data into the bottom tier from operational databases.Data feeding interface called gateways.
  Common gateways are: jdbc, odbc.
- The middle tier is an **OLAP server** that is typically implemented using either (1) a relational OLAP (ROLAP) or (2) a multidimensional OLAP (MOLAP) model.
- The top tier contains query and **reporting tools,analysis tools**, and/or data mining tools (e.g., trend analysis & prediction).

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models: (3 models)

I. Enterprise Warehouse

- It collects all of the information of the the entire organization.
- Contains detailed data as well as summarized data.
- Size ranges from gigabyte to terabyte or more.
- Requires traditional mainframes, computer superservers, or parallel architecture platforms for implementation.
- It requires extensive business modeling.
- It takes years to design and build.

Data Warehouse Models (Cont.)

II. Data Mart

- Contains a subset of corporate-wide data.
- The scope is confined to specific selected subjects and group of users.
- Implemented on low-cost departmental servers (UNIX or Win)
- Short time implementation cycle, measured in weeks.
- It may be **Independent**: sourced from data captured from one or more operational systems or external information providers. Or it can be **Dependent** data marts are sourced directly from enterprise data warehouses

Data Warehouse Models (Cont.)

II. Data Mart

- Contains a subset of corporate-wide data.
- The scope is confined to specific selected subjects and group of users.
- Implemented on low-cost departmental servers (UNIX or Win)
- Short time implementation cycle, measured in weeks.
- It may be **Independent**: sourced from data captured from one or more operational systems or external information providers. Or it can be **Dependent** data marts are sourced directly from enterprise data warehouses

Data Warehouse Models (Cont.)

II. Data Mart

- Contains a subset of corporate-wide data.
- The scope is confined to specific selected subjects and group of users.
- Implemented on low-cost departmental servers (UNIX or Win)
- Short time implementation cycle, measured in weeks.
- It may be **Independent**: sourced from data captured from one or more operational systems or external information providers. Or it can be **Dependent** data marts are sourced directly from enterprise data warehouses

Data Warehouse Models (Cont.)

II. Data Mart

- Contains a subset of corporate-wide data.
- The scope is confined to specific selected subjects and group of users.
- Implemented on low-cost departmental servers (UNIX or Win)
- Short time implementation cycle, measured in weeks.
- It may be **Independent**: sourced from data captured from one or more operational systems or external information providers. Or it can be **Dependent** data marts are sourced directly from enterprise data warehouses

Data Warehouse Models (Cont.)

II. Data Mart

- Contains a subset of corporate-wide data.
- The scope is confined to specific selected subjects and group of users.
- Implemented on low-cost departmental servers (UNIX or Win)
- Short time implementation cycle, measured in weeks.
- It may be **Independent**: sourced from data captured from one or more operational systems or external information providers. Or it can be **Dependent** data marts are sourced directly from enterprise data warehouses

Data Warehouse Models (Cont.2)

III. Virtual warehouse

- A virtual warehouse is a set of views over operational databases.
- **Materialized views** are used for its implementation

Data Warehouse Models (Cont.2)

III. Virtual warehouse

- A virtual warehouse is a set of views over operational databases.
- **Materialized views** are used for its implementation

Extraction, Transformation, and Loading (Life Cycle of DWH)

Following sequential steps:

1. **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
2. **Data cleaning**, which detects errors in the data and rectifies them when possible.
3. **Data transformation**, which converts data from legacy or host format to warehouse format.
4. **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
5. **Refresh**, which propagates the updates from the data sources to the warehouse.

Extraction, Transformation, and Loading (Life Cycle of DWH)

Following sequential steps:

1. **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
2. **Data cleaning**, which detects errors in the data and rectifies them when possible.
3. **Data transformation**, which converts data from legacy or host format to warehouse format.
4. **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
5. **Refresh**, which propagates the updates from the data sources to the warehouse.

Extraction, Transformation, and Loading (Life Cycle of DWH)

Following sequential steps:

1. **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
2. **Data cleaning**, which detects errors in the data and rectifies them when possible.
3. **Data transformation**, which converts data from legacy or host format to warehouse format.
4. **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
5. **Refresh**, which propagates the updates from the data sources to the warehouse.

Extraction, Transformation, and Loading (Life Cycle of DWH)

Following sequential steps:

1. **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
2. **Data cleaning**, which detects errors in the data and rectifies them when possible.
3. **Data transformation**, which converts data from legacy or host format to warehouse format.
4. **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
5. **Refresh**, which propagates the updates from the data sources to the warehouse.

Extraction, Transformation, and Loading (Life Cycle of DWH)

Following sequential steps:

1. **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
2. **Data cleaning**, which detects errors in the data and rectifies them when possible.
3. **Data transformation**, which converts data from legacy or host format to warehouse format.
4. **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
5. **Refresh**, which propagates the updates from the data sources to the warehouse.

Metadata Repository

Metadata

Metadata are **data about data**. Resides **at the bottom tier** of the architecture. Metadata are created for the **data names and definitions** of the given warehouse. **Additional metadata** are created and captured for timestamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.

Information Contained in a Metadata Repository

- **A description of the data warehouse structure**, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- **Operational metadata**, which include **data lineage** (history of migrated data and the sequence of transformations applied to it), **currency of data** (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The **algorithms used for summarization**, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- **Mapping from the operational environment to the data warehouse**, which includes **source databases** and their contents, **gateway descriptions**, data partitions, data extraction, cleaning, **transformation rules and defaults**, **data refresh and purging rules**, and security (user authorization and access control).

Information Contained in a Metadata Repository

- **A description of the data warehouse structure**, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- **Operational metadata**, which include **data lineage** (history of migrated data and the sequence of transformations applied to it), **currency of data** (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The **algorithms used for summarization**, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- **Mapping from the operational environment to the data warehouse**, which includes **source databases** and their contents, **gateway descriptions**, data partitions, data extraction, cleaning, **transformation rules and defaults**, **data refresh and purging rules**, and security (user authorization and access control).

Information Contained in a Metadata Repository

- **A description of the data warehouse structure**, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- **Operational metadata**, which include **data lineage** (history of migrated data and the sequence of transformations applied to it), **currency of data** (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The **algorithms used for summarization**, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- **Mapping from the operational environment to the data warehouse**, which includes **source databases** and their contents, **gateway descriptions**, data partitions, data extraction, cleaning, **transformation rules and defaults**, **data refresh and purging rules**, and security (user authorization and access control).

Information Contained in a Metadata Repository

- **A description of the data warehouse structure**, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- **Operational metadata**, which include **data lineage** (history of migrated data and the sequence of transformations applied to it), **currency of data** (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The **algorithms used for summarization**, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- **Mapping from the operational environment to the data warehouse**, which includes **source databases** and their contents, **gateway descriptions**, data partitions, data extraction, cleaning, **transformation rules and defaults**, **data refresh and purging rules**, and security (user authorization and access control).

Multidimensional Data Model: Cube

A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by **dimensions** and **facts**.

Cube: Dimension

Dimension

- Dimensions are the **perspectives** or entities with respect to which an organization wants to keep records. Example: Sales data warehouse in order to keep records of the store's sales **with respect to** the dimensions time, item, branch, and location.
- Dimension is connected with its **dimension table**, which further describes the dimension. Example: Dimension branch $\Longrightarrow$ dimension table **with further attributes**: branch location, branch code, physical size and so on..

Cube: Dimension

Dimension

- Dimensions are the **perspectives** or entities with respect to which an organization wants to keep records. Example: Sales data warehouse in order to keep records of the store's sales **with respect to** the dimensions time, item, branch, and location.
- Dimension is connected with its **dimension table**, which further describes the dimension. Example: Dimension branch $\Longrightarrow$ dimension table **with further attributes**: branch location, branch code, physical size and so on..

Cube: Fact

Fact

- A multidimensional data model is typically organized around a **central theme** represented by a **fact table**.
- Facts are **numeric measures**. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.
- Example: dollars sold, units sold, units rejected.

Cube: Fact

Fact

- A multidimensional data model is typically organized around a **central theme** represented by a **fact table**.
- Facts are **numeric measures**. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.
- Example: dollars sold, units sold, units rejected.

Cube: Fact

Fact

- A multidimensional data model is typically organized around a **central theme** represented by a **fact table**.
- Facts are **numeric measures**. The fact table contains the names of the facts, or measures, as well as keys to each of the related dimension tables.
- Example: dollars sold, units sold, units rejected.

Cube by an example

- Normally we **think cubes as 3-D geometric structures**, in data warehousing the data cube is **n-dimensional**.
- Lets start at a simple **2-D** (**Items** sold per **Quarter**)data cube

Table 4.2 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

	location = "Vancouver"			
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Cube by an example

- Normally we **think cubes as 3-D geometric structures**, in data warehousing the data cube is **n-dimensional**.
- Lets start at a simple **2-D** (**Items** sold per **Quarter**)data cube

Table 4.2 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

location = "Vancouver"				
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Cube By Example Cont.2

- Lets **add one more dimension** (i.e. **location**) : view the data according to (1) time and (2) item, as well as (3) location. So, it looks like:

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				*location* = "New York"				*location* = "Toronto"				*location* = "Vancouver"			
	item				*item*				*item*				*item*			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Cube By Example Cont.2

- Lets **add one more dimension** (i.e. **location**) : view the data according to (1) time and (2) item, as well as (3) location. So, it looks like:

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				*location* = "New York"				*location* = "Toronto"				*location* = "Vancouver"			
	item				*item*				*item*				*item*			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Cube By Example Cont.2

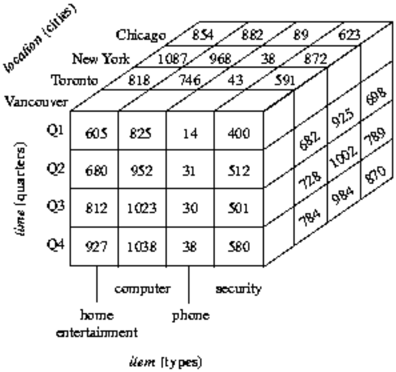
- Lets **add one more dimension** (i.e. **location**) : view the data according to (1) time and (2) item, as well as (3) location. So, it looks like:

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				*location* = "New York"				*location* = "Toronto"				*location* = "Vancouver"			
	item				*item*				*item*				*item*			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

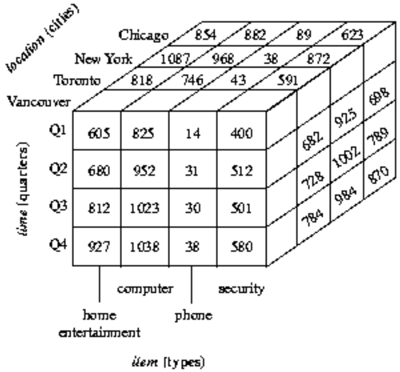
Cube By Example Cont.3

Conceptually, we may also represent the same data in the form of a 3-D data cube, as in following Figure:



Cube By Example Cont.3

Conceptually, we may also represent the same data in the form of a 3-D data cube, as in following Figure:



Cube By Example Cont.4

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				***location*** = "New York"				***location*** = "Toronto"				***location*** = "Vancouver"			
	item				***item***				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

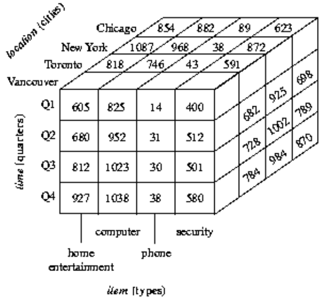


Figure 4.3 A 3 D data cube representation of the data in Table 4.3, according to *tim*

Cube By Example Cont.4

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				***location*** = "New York"				***location*** = "Toronto"				***location*** = "Vancouver"			
	item				***item***				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

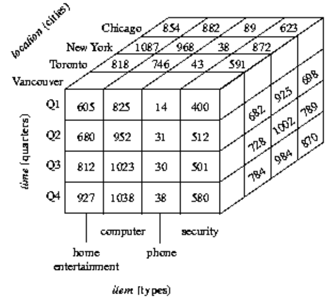


Figure 4.3 A 3 D data cube representation of the data in Table 4.3, according to *tim*

Cube by Example Cont.5

- **Add another dimension** (i.e. **supplier**). Viewing things in 4-D becomes hard. However, we can think of a **4-D cube as** being a **series of 3-D cubes**, as shown in Figure:

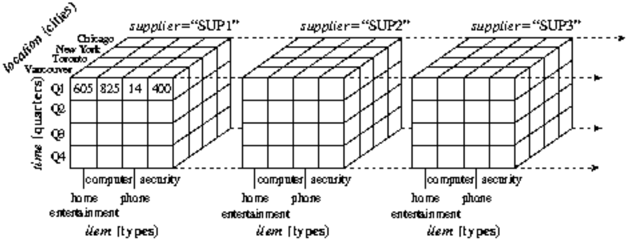


Figure 4.4 A 4-D data cube representation of sales data, according to *time, item, location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of

Cube by Example Cont.5

- **Add another dimension** (i.e. **supplier**). Viewing things in 4-D becomes hard. However, we can think of a **4-D cube as** being a **series of 3-D cubes**, as shown in Figure:

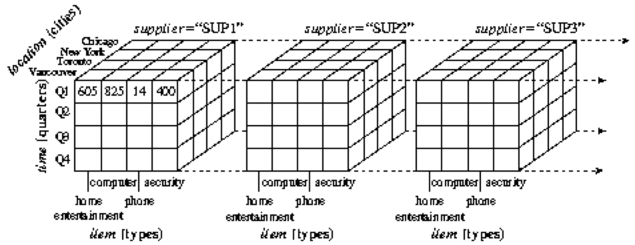


Figure 4.4 A 4 D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of

Cube by Example Cont.5

- **Add another dimension** (i.e. **supplier**). Viewing things in 4-D becomes hard. However, we can think of a **4-D cube as** being a **series of 3-D cubes**, as shown in Figure:

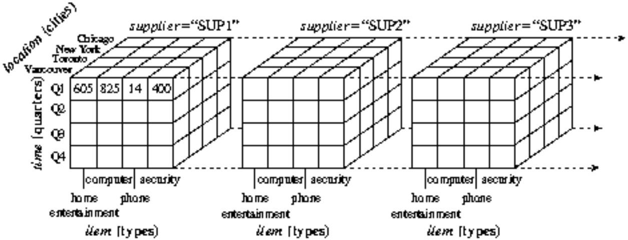


Figure 4.4 A 4 D data cube representation of sales data, according to *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of

Oracle New SQL operator: Cube

- CUBE enables a SELECT statement to calculate subtotals for **all possible combinations** of a group of dimensions.
- **Example:** Dept,Desig,count(*)Total
  from emp group by cube(Dept,Desig)
  order by Dept,Desig;
- Output (Partial):

Dept	Desg	Total
10	Asst. Manager	2
10	Manager	3
10		5
...	..	...

Cuboid

- So far we have seen, data at **different degrees of summarization**.
- In data warehousing literature such **graphical presentation** is called **cuboid**.
- Given a set of dimensions, we can **generate a cuboid** for **each of the possible subsets of the given dimensions**.
- The result would **form a lattice of cuboids**, each showing the **data at a different level of summarization** (or group-by clause in SQL).

Cuboid

- So far we have seen, data at **different degrees of summarization**.
- In data warehousing literature such **graphical presentation** is called **cuboid**.
- Given a set of dimensions, we can **generate a cuboid** for **each of the possible subsets of the given dimensions**.
- The result would **form a lattice of cuboids**, each showing the **data at a different level of summarization** (or group-by clause in SQL).

Cuboid

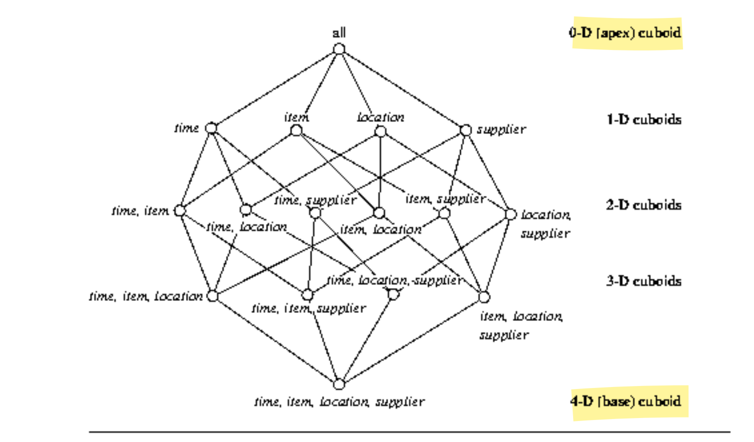
- So far we have seen, data at **different degrees of summarization**.
- In data warehousing literature such **graphical presentation** is called **cuboid**.
- Given a set of dimensions, we can **generate a cuboid** for **each of the possible subsets of the given dimensions**.
- The result would **form a lattice of cuboids**, each showing the **data at a different level of summarization** (or group-by clause in SQL).

Cuboid

- So far we have seen, data at **different degrees of summarization**.
- In data warehousing literature such **graphical presentation** is called **cuboid**.
- Given a set of dimensions, we can **generate a cuboid** for **each of the possible subsets of the given dimensions**.
- The result would **form a lattice of cuboids**, each showing the **data at a different level of summarization** (or group-by clause in SQL).

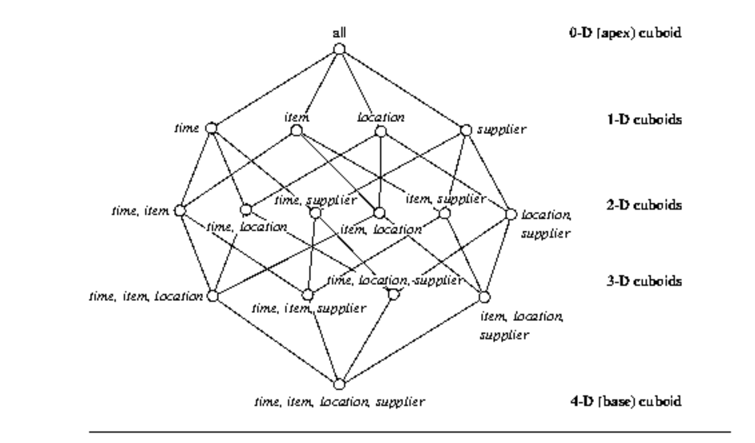
Cuboid (Con.)

- We have **4 dimensions**: time, item, location and supplier



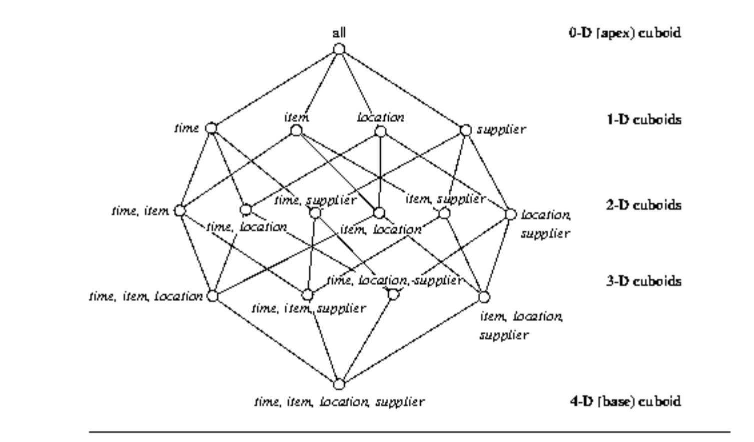
Cuboid (Con.)

- We have **4 dimensions**: time, item, location and supplier

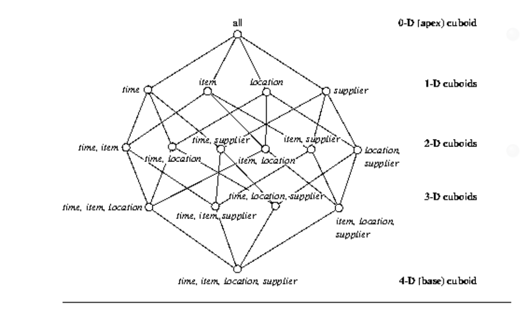


Cuboid (Con.)

- We have **4 dimensions**: time, item, location and supplier

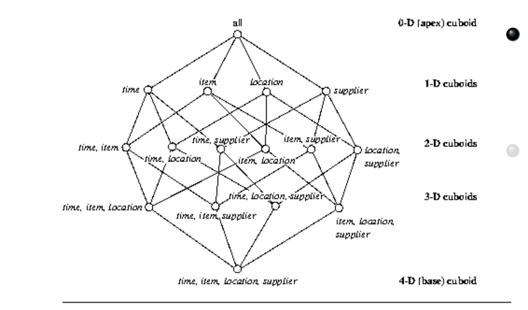


Cuboid (Cont.)



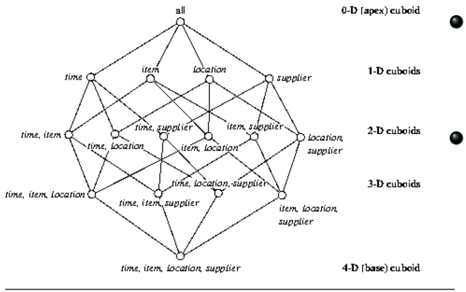
- The cuboid that holds the **lowest level** of summarization is called the **base cuboid**. (where clause..**and-ed by all dimensions**)
- The 0-D cuboid, which holds the **highest level of summarization**, is called the **apex cuboid**. In the **example**, this is the total sales, or dollars sold, summarized over all four dimensions. The apex cuboid is typically **denoted by all**. (where clause..**No where clause...at all**)

Cuboid (Cont.)



- The cuboid that holds the **lowest level** of summarization is called the **base cuboid**. (where clause..**and-ed by all dimensions**)
- The 0-D cuboid, which holds the **highest level of summarization**, is called the **apex cuboid**. In the **example**, this is the total sales, or dollars sold, summarized over all four dimensions. The apex cuboid is typically **denoted by all**. (where clause..**No where clause...at all**)

Cuboid (Cont.)



- The cuboid that holds the **lowest level** of summarization is called the **base cuboid**. (where clause..**and-ed by all dimensions**)
- The 0-D cuboid, which holds the **highest level of summarization**, is called the **apex cuboid**. In the **example**, this is the total sales, or dollars sold, summarized over all four dimensions. The apex cuboid is typically **denoted by all**. (where clause..**No where clause...at all**)

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a concise, subject-oriented schema.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a **concise, subject-oriented schema**.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a **concise, subject-oriented schema**.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a **concise, subject-oriented schema**.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a **concise, subject-oriented schema**.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Data Warehouse Models

ER Model and OLAP

The ER data model is commonly used for for on-line transaction processing.

On the other hand, a data warehouse requires a **concise, subject-oriented schema**.

3 most popular forms:

1. star schema
2. snowflake schema and
3. fact constellation schema

Star Schema Model

The **most common modeling paradigm** is the star schema.

One fact table and a number of dimension tables

It contains:

- A large central table **(fact table)** containing the bulk of the data, **with no redundancy**.
- A set of smaller attendant tables **(dimension tables)**, one for each dimension.
- The model often may introduce **redundancy**. Because of only one single table for each dimension.(A master– detial would solve it..later)

Star Schema Model

The **most common modeling paradigm** is the star schema.

One fact table and a number of dimension tables

It contains:

- A large central table **(fact table)** containing the bulk of the data, **with no redundancy**.
- A set of smaller attendant tables **(dimension tables)**, one for each dimension.
- The model often may introduce **redundancy**. Because of only one single table for each dimension.(A master– detial would solve it..later)

Star Schema Model

The **most common modeling paradigm** is the star schema.

One fact table and a number of dimension tables

It contains:

- A large central table **(fact table)** containing the bulk of the data, **with no redundancy**
- A set of smaller attendant tables **(dimension tables)**, one for each dimension.
- The model often may introduce **redundancy.** Because of only one single table for each dimension.(A master- detial would solve it..later)

Star Schema Model

The **most common modeling paradigm** is the star schema.

One fact table and a number of dimension tables

It contains:

- A large central table **(fact table)** containing the bulk of the data, **with no redundancy**
- A set of smaller attendant tables **(dimension tables)**, one for each dimension.
- The model often may introduce **redundancy.** Because of only one single table for each dimension.(A master- detial would solve it..later)

Star Schema Model

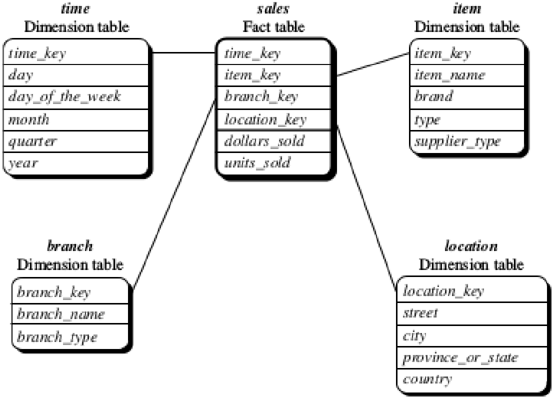
The **most common modeling paradigm** is the star schema.

One fact table and a number of dimension tables

It contains:

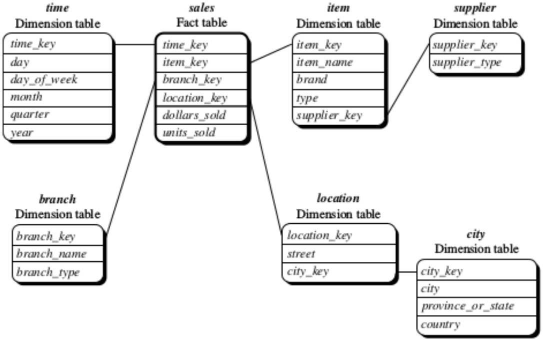
- A large central table **(fact table)** containing the bulk of the data, **with no redundancy**
- A set of smaller attendant tables **(dimension tables)**, one for each dimension.
- The model often may introduce **redundancy.** Because of only one single table for each dimension.(A master– detial would solve it..later)

Star Schema Model: Example



Snowflake schema

It is a variant of the star schema model, where **some dimension tables are normalized**, thereby further splitting the data into additional tables.



Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- Disadvantage:
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- Disadvantage:
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- **Disadvantage:**
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- **Disadvantage:**
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- **Disadvantage:**
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- **Disadvantage:**
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Snowflake schema

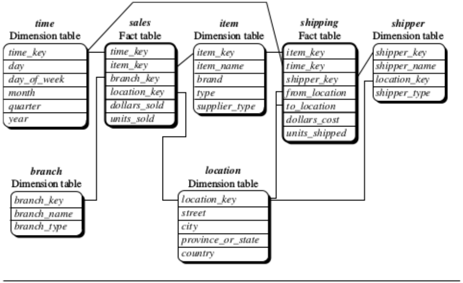
- **Advantage:** Dimension tables of the snowflake model may be kept in **normalized** form **to reduce redundancies**.
- **Disadvantage:**
  - However, this **space savings is negligible** in comparison to the typical magnitude of the fact table.
  - Furthermore, the snowflake structure can reduce the effectiveness of browsing, since **more joins will be needed to execute a query**. Consequently, the system performance may be adversely impacted.

Therefore..

Although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.

Fact constellation

Sophisticated applications may require **multiple fact tables to share dimension tables**. This kind of schema can be **viewed as a collection of stars**, and hence is called a galaxy schema or a fact constellation.



Have a look on these topics:

- Dimensions: The Role of Concept Hierarchies
- Measures: Their Categorization and Computation

Typical OLAP Operations

- **Roll-up:** Performs **aggregation** on a data cube either by **climbing up a concept hierarchy** for a dimension or by dimension reduction.
- **Drill-down:** Drill-down is the **reverse of roll-up**. It **navigates from less detailed data to more detailed** data. Drill-down can be realized **by either stepping down a concept hierarchy** for a dimension or **introducing additional dimensions**.
- **Slice and dice:** The slice operation performs a **selection on one dimension** of the given cube, resulting in a subcube. The dice operation defines a subcube by performing a **selection on two or more dimensions**.
- **Pivot (rotate):** Pivot (also called **rotate**) is a visualization operation that **rotates the data axes** in view to provide an **alternative data presentation**.

Typical OLAP Operations

- **Roll-up:** Performs **aggregation** on a data cube either by **climbing up a concept hierarchy** for a dimension or by dimension reduction.
- **Drill-down:** Drill-down is the **reverse of roll-up**. It **navigates from less detailed data to more detailed** data. Drill-down can be realized **by either stepping down a concept hierarchy** for a dimension or **introducing additional dimensions**.
- **Slice and dice:** The slice operation performs a **selection on one dimension** of the given cube, resulting in a subcube. The dice operation defines a subcube by performing a **selection on two or more dimensions**.
- **Pivot (rotate):** Pivot (also called **rotate**) is a visualization operation that **rotates the data axes** in view to provide an **alternative data presentation**.

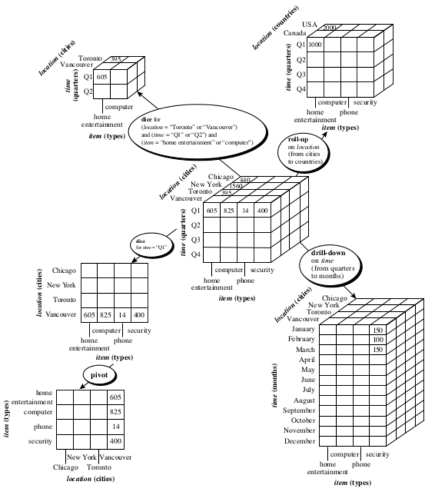
Typical OLAP Operations

- **Roll-up:** Performs **aggregation** on a data cube either by **climbing up a concept hierarchy** for a dimension or by dimension reduction.
- **Drill-down:** Drill-down is the **reverse of roll-up**. It **navigates from less detailed data to more detailed** data. Drill-down can be realized **by either stepping down a concept hierarchy** for a dimension or **introducing additional dimensions**.
- **Slice and dice:** The slice operation performs a **selection on one dimension** of the given cube, resulting in a subcube. The dice operation defines a subcube by performing a **selection on two or more dimensions**.
- **Pivot (rotate):** Pivot (also called **rotate**) is a visualization operation that **rotates the data axes** in view to provide an **alternative data presentation**.

Typical OLAP Operations

- **Roll-up:** Performs **aggregation** on a data cube either by **climbing up a concept hierarchy** for a dimension or by dimension reduction.
- **Drill-down:** Drill-down is the **reverse of roll-up**. It **navigates from less detailed data to more detailed** data. Drill-down can be realized **by either stepping down a concept hierarchy** for a dimension or **introducing additional dimensions**.
- **Slice and dice:** The slice operation performs a **selection on one dimension** of the given cube, resulting in a subcube. The dice operation defines a subcube by performing a **selection on two or more dimensions**.
- **Pivot (rotate):** Pivot (also called **rotate**) is a visualization operation that **rotates the data axes** in view to provide an **alternative data presentation**.

OLAP Operations (Cont.)



Thank You.