Lecture 3
Data Warehouse and OLAP Technology

CA4010: Data Warehousing and Data Mining
2019/2020 Semester 1

Dr. Mark Roantree
Dublin City University

Agenda

What is a Data Warehouse?

- A decision support database that is maintained separately from the organisation's operational database.
- Support information processing by providing a solid platform of consolidated, historical data for analysis.
- Data Warehousing: The process of constructing and using data warehouses.

Inmon's Definition

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

Data Warehouse: Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Inmon's Definition

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

Data Warehouse: Integrated

- Constructed by integrating multiple, heterogeneous data sources
  relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  Ensure consistency in naming conventions, encoding structures, attribute measures, among different data sources
  Hotel price: currency, tax, breakfast covered, etc.
  When data is moved to the warehouse, it is converted.

Inmon's Definition

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

Data Warehouse: Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain *time element*

Inmon's Definition

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

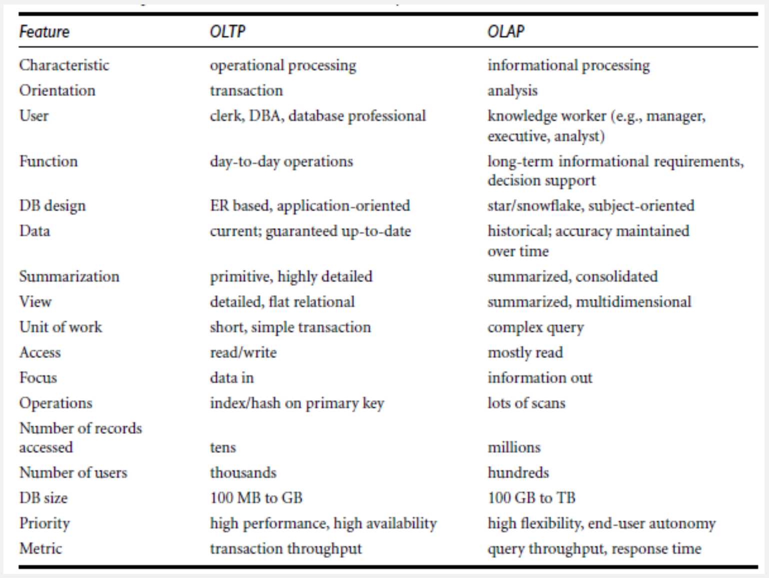
Data Warehouse: Non-volatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing: initial loading of data and access of data

Inmon's Definition

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

OLTP vs. OLAP



Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements, decision support
DB design	ER based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

Figure 1: OLTP vs. OLAP: A Comparison

Properties of a Warehouse: Data Engineering

- Integrated data
- Detailed and summarised data
- Historical data
- Metadata

Integrated Data

- Integrated data allows the miner to easily examine very high volumes of data.
- Without integrated data, too much time is spent cleaning and transforming data before data mining can commence effectively.
- Keys must be reconstituted;
  encoded values reconciled;
  data structures standardized;
  and many more tasks before data mining.
- All of these tasks have been completed *before* the data warehouse.

Detailed and Summarised Data

- Detailed data is necessary when the miner wishes to examine data in its most *granular* form.
- Very low levels of detail hide important patterns that require the study of very low level (detailed) data.
- By contrast, summarised data ensures the reuse of a prior analysis (using a pre-existing *cube*).
- Summarized data ensures the miner can build on the work of others rather than build everything from the ground up.

Historical Data

- Historical data is important to the miner because important knowledge can be hidden there.
- A miner working only with very current information cannot detect trends and long-term patterns of behavior.
- Historical information is crucial to understanding the seasonality of business and the larger cycles of business to which every corporation is subject.

Metadata

- Metadata serves as a *road map* to the miner, who uses metadata to describe the *context* of information.
- When examining information *over time*, context becomes as relevant as content.
- In other words, raw data becomes very difficult for the data miner to work with when there is no explanation for the *meaning* of the data.
- The data warehouse provides a platform for successful and efficient exploration of the world of data, with the miner exploiting metadata for more powerful analyses.

Why a Separate Data Warehouse?

- High performance for both systems
  - DBMS tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data
  - missing data: Decision support requires historical data which operational DBs do not typically maintain
  - data consolidation: DS requires consolidation (aggregation, summarisation) of data from heterogeneous sources
  - data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

Data Warehouse: A Multi-Tiered Architecture

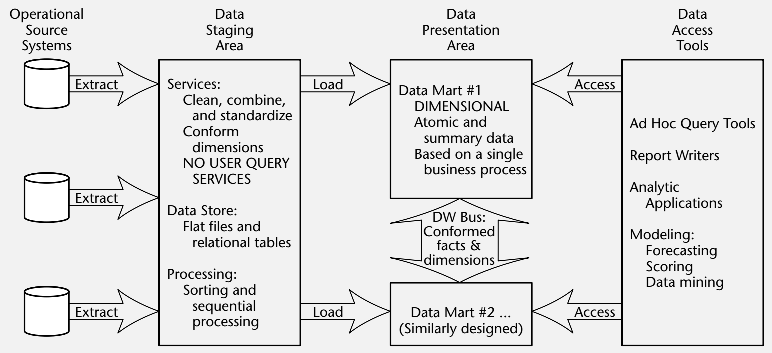

Operational Source Systems
Data Staging Area
Data Presentation Area
Data Access Tools
Extract
Extract
Extract
Services: Clean, combine, and standardize Conform dimensions NO USER QUERY SERVICES
Data Store: Flat files and relational tables
Processing: Sorting and sequential processing
Load
Load
Data Mart #1 DIMENSIONAL Atomic and summary data Based on a single business process
DW Bus: Conformed facts & dimensions
Data Mart #2 ... (Similarly designed)
Access
Access
Ad Hoc Query Tools
Report Writers
Analytic Applications
Modeling: Forecasting Scoring Data mining

Figure 2: ETL Architecture

Extraction, Transformation, and Loading (ETL)

- **Data extraction**: get data from multiple, heterogeneous, and external sources
- **Data cleaning**: detect errors in the data and rectify them when possible
- **Data transformation**: convert data from legacy or host format to warehouse format
- **Load**: sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions
- **Refresh**: propagate the updates from the data sources to the warehouse

Metadata Repository

Metadata is the data defining warehouse objects. It stores:

- Description of the data warehouse schema, view, dimensions, hierarchies, derived data definitions, data mart locations and contents
- Operational meta-data: data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
- The algorithms used for summarisation
- The mapping from operational databases to the data warehouse
- Data related to system performance: warehouse schema, view and derived data definitions
- Business data: business terms and definitions, ownership of data, charging policies

From Tables and Spreadsheets to Data Cubes

- A data warehouse is based on a **multidimensional** data model which views data in the form of a **data cube**.
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as amount_sold) and keys to each of the related dimension tables
- In data warehousing, an *n*-D base cube is called a **base cuboid**. The top most *0*-D cuboid, which holds the highest-level of summarisation, is called the **apex cuboid**.
- The *lattice* of cuboids forms a data cube.

Cube: A Lattice of Cuboids

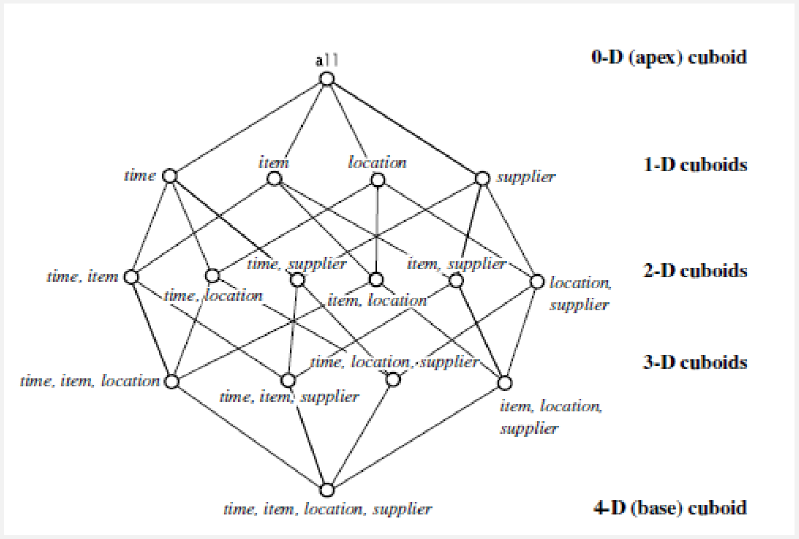


Figure 3: 4-D Data Cube Representation

Conceptual Modeling of Data Warehouses

Modeling data warehouses: **dimensions** & **measures**

- **Star schema**: A *fact* table in the middle *connected* to a set of dimension tables.
- **Snowflake schema**: A star schema refinement where a dimensional hierarchy is *normalised* into a set of smaller dimension tables, forming a shape similar to a snowflake.
- **Fact constellations**: Multiple fact tables *share dimensions*, viewed as a collection of stars, called a *galaxy* schema or *fact constellation*.

Star Schema

- The most common modeling paradigm is the **star schema**, in which the data warehouse contains
  1. a large central table (fact table) containing the bulk of the data, with no redundancy, and
  2. a set of smaller attendant tables (dimension tables), one for each dimension.
- The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.

Star Schema for a Sales Data Warehouse

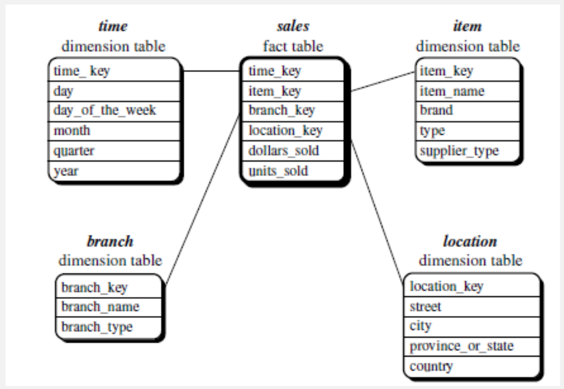


Figure 4: AllElectronics Star Schema

- Sales are considered along four dimensions: *time*, *item*, *branch*, and *location*.
- The schema contains a central **fact table** for *sales* that contains keys to each of the four dimensions.
- Contains two measures: *dollars sold* and *units sold*.

Star Schema Model

- Each dimension is represented by a single table, containing a set of attributes.
- The *location* dimension table contains the attribute set *{location key, street, city, province or state, country}*.
- This constraint may introduce some redundancy: *Vancouver* and *Victoria* are both cities in the *Canadian* province of *British Columbia*.
- This will create redundancy among the attribute province: *British Columbia* will be repeated for each (BC) city.
- Attributes within a dimension may form either a hierarchy (total ordering) or a lattice (partial ordering).

Snowflake Schema for a Sales Data Warehouse

The **snowflake schema** is a variant of the star schema model, where some dimension tables are normalised splitting dimensional data into additional tables.

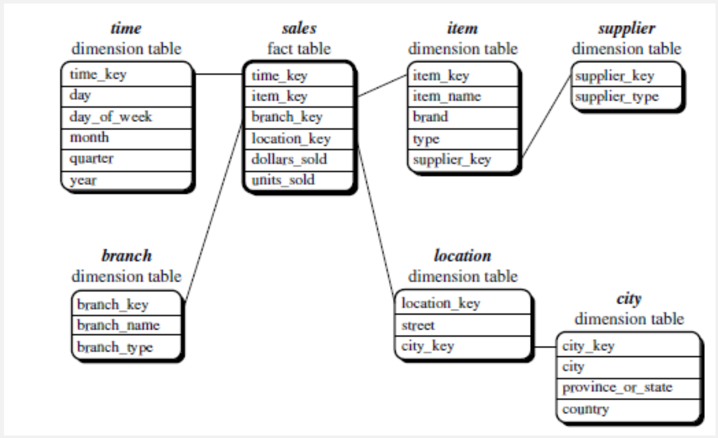


Figure 5: Snowflake Schema for *item* and *location*

Snowflake Schema

- The major difference between the *snowflake* and *star* schema models is that snowflake dimensions may use normalisation to reduce redundancies.
- However, this saving of space is negligible in comparison to the typical magnitude of the fact table.
- Furthermore, the *snowflake* structure can reduce the performance, since more joins will be needed to execute a query.
- 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.
- The schema in figure 6 specifies two fact tables: *sales* and *shipping*.

Fact Constellation Schema for Sales and Shipping

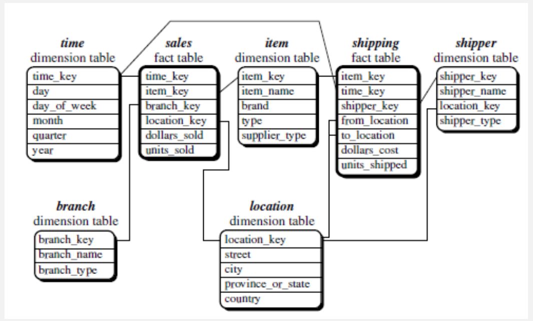


Figure 6: Sample Fact Constellation Schema

- Shipping has 5 dimensions (keys): *item key, time key, shipper key, from location*, and *to location*, and two measures: *dollars cost* and *units shipped*.
- A *fact constellation* schema allows dimension tables to be shared between fact tables.
- For example, dimensions for *time, item,* and *location* are shared by both the *sales* and *shipping* facts.

Concept Hierarchy

- A **concept hierarchy** defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.
- Consider a concept hierarchy for the dimension location: City values for location include *Vancouver*, *Toronto*, *New York*, and *Chicago*.
- Each city can be mapped to its province or state.
- Provinces and states can in turn be mapped to their country.
- These mappings form a concept hierarchy for the dimension location, mapping a set of low-level concepts (cities) to higher-level, more abstract concepts (countries).

Location Concept Hierarchy

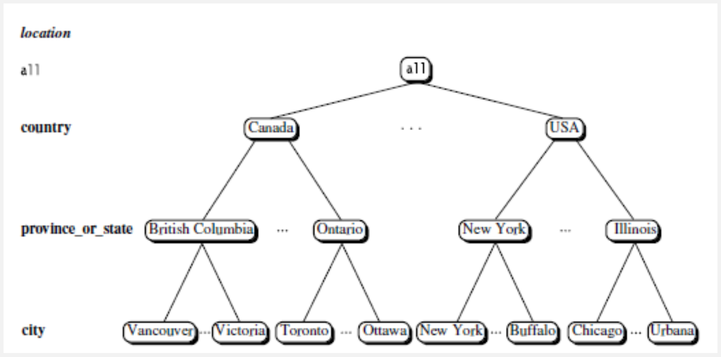


Figure 7: Location Concept Hierarchy

Multidimensional Data

Sales volume as a function of product, month, and region

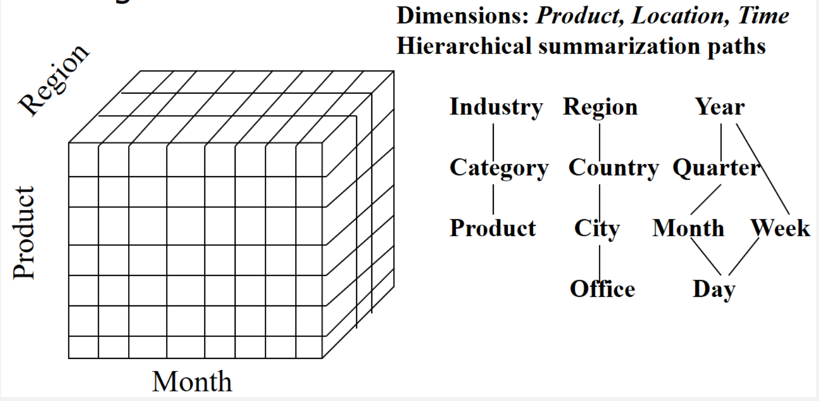


Figure 8: Sales by Product, Time, Geo-Location

Cube: Multidimensional Construct

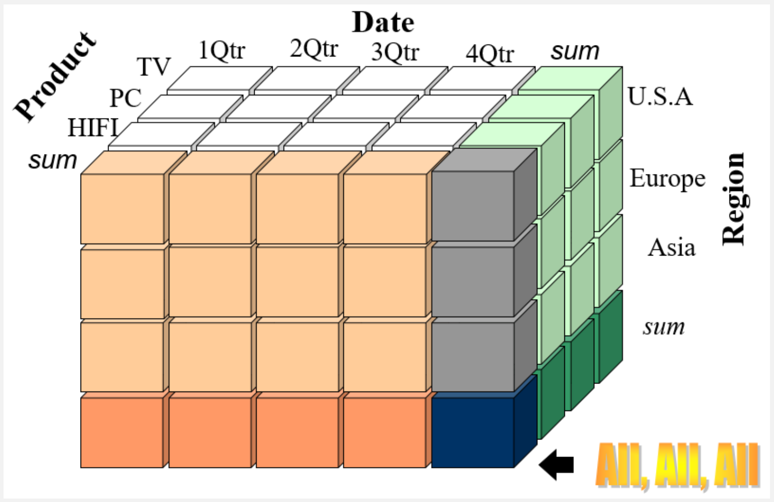
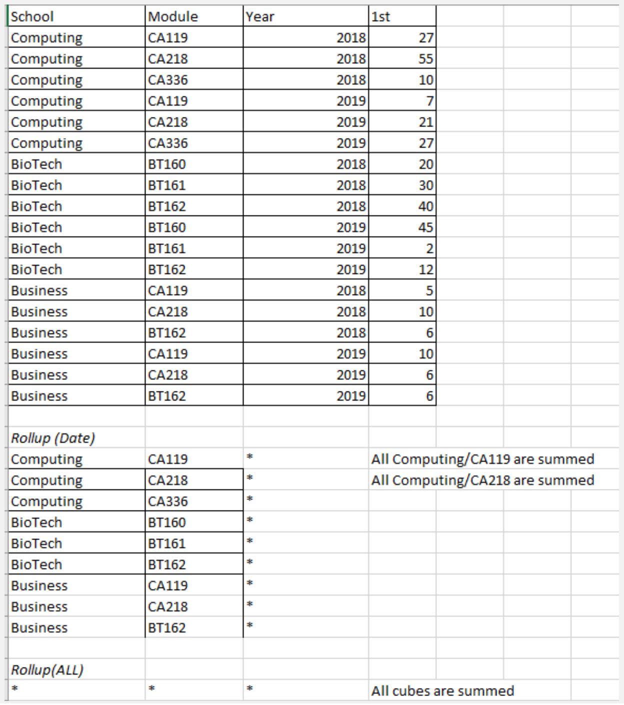


Figure 9: Cube Components

OLAP Operations

- **Roll up**: summarise data by climbing up the hierarchy or by dimension reduction.
- **Drill down**: reverse of roll-up from higher level summary to more detail, or introducing new dimensions.
- **Slice and dice**: project and select.
- **Pivot** (rotate): reorient the cube, visualisation, 3D to series of 2D planes.
- **Drill across**: involving (across) more than one fact table.
- **drill through**: through the bottom level of the cube to its back-end relational tables (using SQL).

Rollup Operation



School	Module	Year	1st
Computing	CA119	2018	27
Computing	CA218	2018	55
Computing	CA336	2018	10
Computing	CA119	2019	7
Computing	CA218	2019	21
Computing	CA336	2019	27
BioTech	BT160	2018	20
BioTech	BT161	2018	30
BioTech	BT162	2018	40
BioTech	BT160	2019	45
BioTech	BT161	2019	2
BioTech	BT162	2019	12
Business	CA119	2018	5
Business	CA218	2018	10
Business	BT162	2018	6
Business	CA119	2019	10
Business	CA218	2019	6
Business	BT162	2019	6
Rollup (Date)			
Computing	CA119	*	All Computing/CA119 are summed
Computing	CA218	*	All Computing/CA218 are summed
Computing	CA336	*	
BioTech	BT160	*	
BioTech	BT161	*	
BioTech	BT162	*	
Business	CA119	*	
Business	CA218	*	
Business	BT162	*	
Rollup(ALL)			
*	*	*	All cubes are summed

Typical OLAP Operations

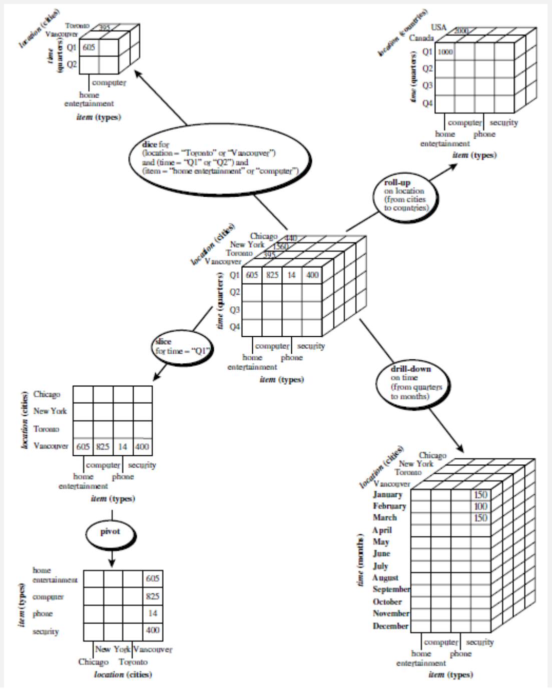


Figure 10: OLAP Operations

Four-Step Dimensional Design Process

1. Select the business process to model.
2. Declare the grain of the business process
3. Choose the dimensions that apply to each fact table row
4. Identify the numeric facts that will populate each fact table row.

Step 1. Select the business process to model.

- A **process** is a natural business activity supported by source data.
- **Business measurement** processes provide the performance measurements users need to analyse in the data warehouse.
- Example business processes:
  How many Female students taking CA707, who live in Munster, are scoring firsts?
  How many frozen products, imported from France, are sold in January in April?
  (these are dimensional queries!)
- Never refer to an organisational business department or function when we talk about business processes.

Consistent Information

- Build a *single dimensional model* to handle orders data rather than building separate models for sales and marketing, who both want access to orders data.
- Focus on business processes (not departments) to deliver *consistent* information more efficiently throughout the organisation.

Step 2. Declare the Grain

- Specify *exactly* what an individual fact table row represents.
- The **grain** conveys the level of detail associated with the fact table measurements.
- It provides the answer to the question: *How do you describe a single row in the fact table?*
  For our 2 examples, **grain is**:
  Every mark for every module for every student
  Every item sold (units), per country of origin, by month, by sales location
- It is virtually impossible to reach closure in step 3 without declaring the grain.

Sample grain declarations

- An individual line item on a customer's retail sales ticket as measured by a scanner device
- A line item on a bill received from a doctor
- An individual boarding pass to get on a flight
- A daily snapshot of the inventory levels for each product in a warehouse
- A monthly snapshot for each bank account

Step 3. Choose the Dimensions (for every measure)

- Dimensions emerge from asking: *How do business people describe the data that results from the business process?*
- Enrich fact tables with a robust set of dimensions representing all possible descriptions.
- Once the grain is known, dimensions can be identified easily.
- For each dimension, list all discrete, text-like attributes to populate each dimension table.
- *What dimensions were used in our 2 examples?*

Step 4. Identify the numeric facts that will populate each fact table row.

- Facts are determined by answering the question: *What are we measuring?*
- All candidate facts in a design must be true to the grain defined in step 2.
- Facts that clearly belong to a different grain *must be in a separate fact table*.
- Typical facts are numeric additive figures such as quantity ordered or dollar cost amount.

Summary

Consider both business users' requirements *and* realities of source data, to make decisions regarding the 4 steps.

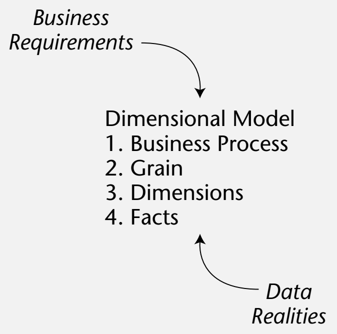


Figure 11: Key input to the four-step dimensional design process

Large Grocery Chain: Case Study

- Business has 100 stores across multiple regions.
- Each store has a full set of departments: grocery, frozen foods, dairy, meat, produce, bakery, floral, and health/beauty aids.
- Each store has roughly 60,000 individual products, called stock keeping units (SKUs).
- About 55,000 of the SKUs come from outside manufacturers and have bar codes imprinted on the product package.

Case Study: Grain

- Bar codes are universal product codes (UPCs).
- UPCs are at the same grain as individual SKUs.
- Each different package variation of a product has a separate UPC and thus, a separate SKU.
- The remaining 5,000 SKUs come from departments such as meat, produce, bakery, or floral.
- These products don't have UPCs, so the grocery chain assigns SKU numbers to them.

Step 1. Select the Business Process

- The first dimensional model built should have **maximum impact**: it should answer the most pressing business questions.
- In our retail case study, management wants to *better understand* customer purchases as captured by the POS system: POS retail sales is selected.
- This data allows them to analyse *what products* are selling, in *which stores*, on *what days*, under what promotional *conditions*.

Step 2. Declare the Grain

- *What level of data detail should be made available in the dimensional model?*
- Tackling data at its lowest, **most atomic grain** makes sense on multiple fronts.

Atomic Data: Advantages

1. Atomic data is **highly dimensional**.
2. The more detailed and atomic the fact measurement, the more things we know for sure (correct, certain).
3. All those things we know for certain *translate into dimensions*.
4. Atomic data provides maximum analytic flexibility because it can be *constrained* and *rolled up* in every way possible.

Higher Grains: Disadvantages

- Declare higher-level grains for a business process that represent an aggregation of the most atomic data.
- However, by selecting a higher-level grain, users are limited to fewer and/or potentially less detailed dimensions.
- The less granular model is immediately vulnerable to unexpected user requests to drill down into the details.
- Users hit an *analytic wall* when not given access to the atomic data.

Step 3. Choose the Dimensions

- Once the grain is chosen, the *date*, *product*, and *store* dimensions become obvious.
- We assume that the calendar date is the date value delivered to us by the POS system.
- Within the framework of the primary dimensions, we consider new dimensions such as the *promotion* under which the product is sold.

- Assume we decide on the following descriptive dimensions: `date`, `product`, `store`, and `promotion`.
- Include the POS transaction ticket number as a special dimension.
- A preliminary schema is shown in figure 12.
- *Do not begin* populating the dimension tables with descriptive attributes: instead complete the final step of the process (keep it simple for now!).

Preliminary Schema

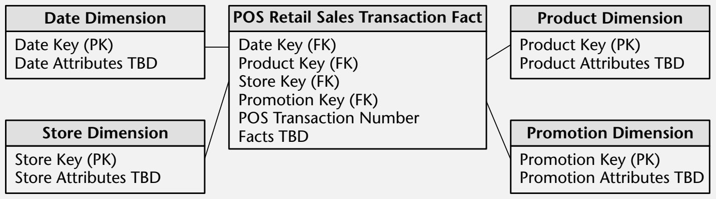


Figure 12: Preliminary retail sales schema

Step 4. Identify the Facts

- There is a firm rule for this step: **the facts must be true to the grain**.
- Specifically, this is the individual line item on the POS transaction.
- The facts collected by the POS system include the sales quantity, per unit sales price, and the sales amount.
- The sales amount is: *quantity by unit price*.

- The current fact table is shown in Figure 13.
- Three of the facts, *sales quantity*, *sales amount*, and *cost amount*, are perfectly additive across all the dimensions.
- We can **slice** and **dice** the fact table as required, and every sum of these three facts is *valid* and *correct*.

Measured facts in the retail sales schema

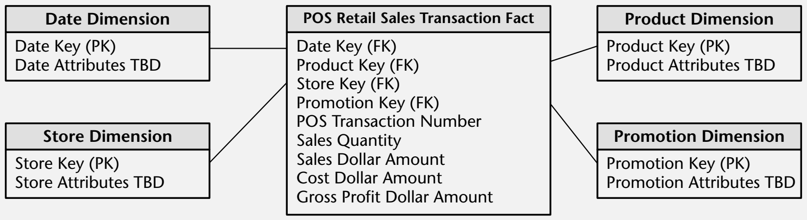


Figure 13: Measured facts in the retail sales schema

Computed Facts

- Compute gross profit by subtracting the *cost amount* from the *sales amount*.
- Although *computed*, this gross profit is also *perfectly additive* across *all* the dimensions.
- One can calculate the *gross profit* of any *combination of products* sold in *any set of stores* on *any set of days*.
- *Should a computed fact be stored physically in the database?*

Computed Facts: Storage Benefits

- Computing and storage of facts eliminates user error.
- Storing them also ensures that all users and their reporting applications refer to gross profit consistently.
- Pre-computation is a form of optimisation.

Non Additive Facts

- The gross margin can be calculated by dividing the gross profit by revenue (*sales amount - cost*).
- Gross margin is a **non-additive fact** because it cannot be summarized along any dimension.

Non Additive Facts: General Rule

- Percentages and ratios, such as gross margin, are non-additive.
- The *numerator* and *denominator* should be stored in the fact table.
- Other attributes (eg. unit price in this case study) will also be non-additive: what's the average price of a piece of fruit? This has no meaning!

3-tier DW Architecture

Data warehouses often adopt a three-tier architecture.

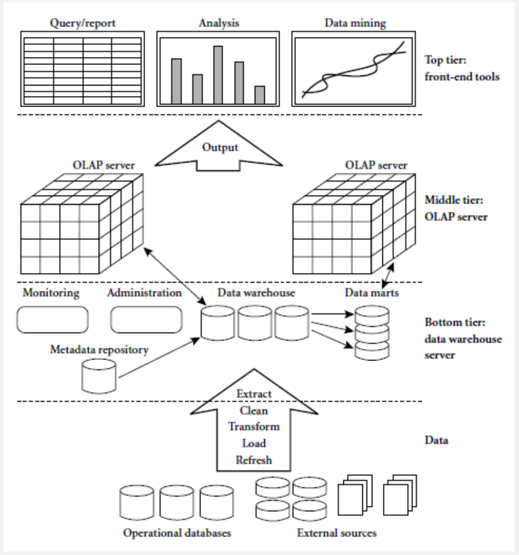


Figure 14: 3-tier DW Architecture

Bottom Tier: DW Server

- The bottom tier is a warehouse database server: generally a relational database system.
- Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources.
- These tools and utilities perform **data extraction**, **cleaning**, and **transformation** (to integrate data), as well as load and refresh functions to update data warehouse.
- The data are extracted using application program interfaces known as **gateways** (SQL programs).
- There is also a metadata repository, which stores information about the data warehouse and its contents.

Middle Tier: OLAP Server

- The middle tier is an OLAP server that is typically implemented using either:
  1. a relational OLAP (ROLAP) model, that is, an extended relational DBMS that maps operations on multidimensional data to standard relational operations or
  2. a multidimensional OLAP (MOLAP) model, that is, a special-purpose server that directly implements multidimensional data and operations.

Top Tier: Front End

- The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools: trend analysis, prediction, etc.
- From the architecture point of view, there are three data warehouse models: the *enterprise warehouse*, the *data mart*, and the *virtual warehouse*.

Enterprise Warehouse

- An **enterprise warehouse** collects all information about subjects from the entire organisation.
- It provides corporate-wide data integration, from internal and external data and is *cross-functional* in scope.
- It contains detailed data and summarised data, and can range in size from a few gigabytes to hundreds of terabytes.
- An enterprise data warehouse may be implemented on traditional mainframes, computer super servers, or parallel architecture platforms.
- It requires extensive business modeling and may take years to design and build.

Data Mart

- A **data mart** contains a subset of corporate-wide data that is of value to a specific group of users.
- The scope is confined to specific selected subjects: a *marketing* data mart may confine subjects to *customer*, *item*, and *sales*.
- The data contained in data marts tend to be summarised.
- The implementation cycle is generally measured in weeks rather than months or years.
- However, it may involve complex integration in the long run if its design and planning were not enterprise-wide.

Data Mart Types

- Depending on the source of data, data marts can be categorized as *independent* or *dependent*.
- **Independent data marts** are sourced from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area.
- **Dependent data marts** are sourced directly from enterprise data warehouses.

Virtual Warehouse

- A **virtual warehouse** is a set of views over operational databases.
- For efficient query processing, only some of the possible summary views may be materialised.
- A virtual warehouse is easy to build but requires excess capacity on operational database servers.

Warehouse Implementation

- Which implementation is best?
- A virtual warehouse has a lot of limitations.
- An enterprise warehouse is very expensive.
- Are there downsides to the data mart approach? Are they avoidable?

Data Warehouse Bus Architecture

- By defining a *standard bus interface* for the data warehouse environment, separate data marts can be implemented by *different* groups at *different* times.
- The separate data marts can be integrated if they adhere to the standard.
- This approach adopts the *data warehouse bus architecture*.

Design Methodology

The bus architecture provides a rational approach to decomposing enterprise warehouse planning.

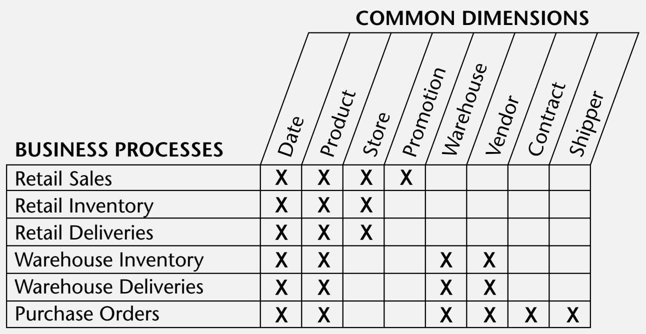
1. Design a **master suite** of standardised dimensions and facts with **uniform interpretation** across the enterprise.
2. This establishes the *data architecture framework*.
3. Build the first (next) data mart that closely adheres to the architecture.
4. Repeat Step 3 until enough data marts exist to deliver an *integrated* enterprise data warehouse.

Benefits: reduce the Size of the Problem

- The bus architecture offers the best of both worlds.
- They have an architectural framework that guides the overall design, but the problem is divided into manageable data mart segments.
- Separate data mart development teams follow the architecture guidelines while working fairly independently and asynchronously.
- If designed coherently, marts share a **uniform architecture** of *conformed* dimensions and facts for seamless integration.

Data Warehouse Bus Matrix

- The rows of the bus matrix correspond to data marts.
- Create rows where sources/processes are different or where a process is too big.



	COMMON DIMENSIONS							
BUSINESS PROCESSES	Date	Product	Store	Promotion	Warehouse	Vendor	Contract	Shipper
Retail Sales	X	X	X	X				
Retail Inventory	X	X	X					
Retail Deliveries	X	X	X					
Warehouse Inventory	X	X			X	X		
Warehouse Deliveries	X	X			X	X		
Purchase Orders	X	X			X	X	X	X

Figure 15: Sample Data Warehouse Bus Matrix

Constructing the Bus Matrix

- The tool used to *create*, *document*, and *communicate* the bus architecture is the **data warehouse bus matrix**.
- Working in tabular fashion, lay out the business processes of the organisation as matrix rows (should be sources of data).
- Matrix rows translate into **data marts** based on the organisation's primary activities.
- Begin with data marts derived from a single primary source, known as *first-level data marts*.

First Level Data Marts

- *Step 1*. Begin with *first-level data marts* as they minimize the risk of failure.
- Most of the overall risk of failure comes from taking on too much of the extract transformation-load (ETL) data staging design and development effort.
- In many cases, first-level data marts provide users with enough interesting data and ROI while the data mart teams focus on more difficult issues.

Multisource Marts

- *Step 2*. Identify more complex multi-source marts
- Referred to as *consolidated data marts* because they typically cross business processes.
- It is prudent to focus on the first-level data marts as dimensional building blocks before tackling the task of consolidating.
- In some cases, the consolidated data mart is a *simple union of data sets* from the first-level data marts.
- Thus, it can often be easier to rebuild the consolidation *within* the data warehouse.

Using the Matrix

- The columns of the matrix represent the common *dimensions* used across the enterprise.
- *Maybe create a comprehensive list of dimensions before filling in the matrix*?
- **X** indicates that the dimension column is related to the business process row.
- Looking across the rows captures the *dimensionality* of each data mart at a glance.
- Visual power: examine the columns that depict the *interaction* between the *data marts* and *common dimensions*.

Matrix Benefits: planning and communication

- Defines the overall data architecture for the warehouse (by laying out rows & columns)
- *What is a high priority dimensions?*: participation in multiple data marts
- Visually: communicate effectively within and across data mart teams.

A Framework: Integration and Interoperability

- It represents poor design to build separate data marts that ignore a *conforming* framework.
- Isolated, independent data marts deliver access to irreconcilable views of the organisation with reports that cannot be compared.
- Independent data marts become legacy implementations, blocking the development of a coherent warehouse environment.

Conformed Dimensions

- Conformed dimensions are either identical or strict mathematical subsets of the most granular, detailed dimension.
- Conformed dimensions have:
  - consistent dimension keys,
  - consistent attribute column names,
  - consistent attribute definitions, and
  - consistent attribute values (which translates into consistent report labels and groupings).
- Non-conforming dimensions give invalid results!

- Conformed dimensions come in several different flavors.
- At the most basic level, conformed dimensions have *identical meaning* with every possible fact table to which they are joined.
- The *date* dimension table connected to the *sales facts* is identical to the *date* dimension table connected to the *inventory facts*.
- In fact, the conformed dimension may be the *same physical table* within the database.
- In general, the date dimensions in both data marts will have: the same number of rows; same key values; same attribute labels; same attribute definitions; and same attribute values.

Conforming and Grain

- Most conformed dimensions are defined naturally at the *most granular level possible*.
- The grain of the *customer* dimension will be the individual customer.
- The grain of the *product* dimension will be the lowest level at which products are tracked in the source systems.
- The grain of the *date* dimension will be the individual day.

Different Grains

- Sometimes dimensions are needed at a rolled-up level of granularity.
- Why? Perhaps the roll-up dimension is required because the fact table *represents aggregated facts* that are associated with *aggregated dimensions*.
- This would be the case if we had a weekly inventory snapshot *in addition* to our daily snapshot.
- In other situations, the facts simply may be generated by another business process at a higher level of granularity.

Conforming by Roll-up

- Assume a sales process captures data at the **atomic** *product level*, while forecasting generates data at the *brand level*.
- It is not possible to share a single product dimension table across the two business process schemas because the granularity is different.
- Product and brand dimensions still conform if the brand table were a *strict subset* of the atomic product table.
- Attributes that are common to both the detailed and rolled-up dimension tables, such as the *brand* and *category* descriptions, should be **labeled**, **defined**, and **valued** identically in both tables, as Figure 16.

- Roll-up dimensions conform to atomic dimension if they are a strict subset of that atomic dimension.

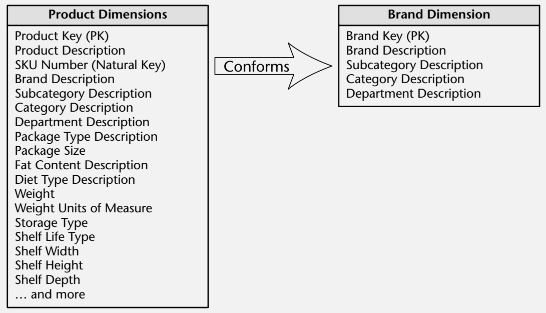


Figure 16: Conforming Rollup Dimension Subsets

- Another case of conformed dimension subsetting occurs when two dimensions are at the same level of detail but one represents only a subset of rows.
- Analysts in the separate businesses may want to view only their subset of the corporate dimension, restricted to the product rows for their business.
- By using a subset of rows, they are not obliged to use the entire product set for the organisation.

Issues with Dimension Row Subsetting

- The fact table joined to this subsetted dimension *must be limited* to the same subset of products.
- An attempt to use a subset dimension while accessing a fact table consisting of the complete product set, may result in unexpected query results.
- Technically, referential integrity would be violated.
- It is necessary to be aware of the potential for user confusion or error with dimension row subsetting.

Fact Table Comparison

- There are three fundamental types of fact tables: **transaction**, **periodic snapshot**, and **accumulating snapshot**.
- These three fact table variations are not totally dissimilar because they share *conformed dimensions*.
- These are the keys to building separate fact tables that can be used together with common, consistent filters and labels.
- While the dimensions are shared, the usage and volume of the three fact tables differ.

Transaction Fact Tables

- Most fundamental view of business's operations is at the *individual transaction level*.
- A row exists in the fact table for a given customer or product *only* if a transaction event occurred.
- On the other hand, a specific customer or product should be linked to *multiple* rows in the fact table as the customer or product is involved in multiple transactions ($1 - to - many$ relationship between dimensions attributes and facts).

- Transaction data fits easily into a dimensional framework.
- The lowest-level data is naturally dimensional, supporting analyses not possible on summarised data (detailed analysis).
- However, analysts cannot survive on transactions alone: too much data, too slow!

Periodic Snapshot Fact Tables

- **Periodic snapshots** are needed to see the *cumulative performance* of the business at regular, predictable time intervals.
- Take a snapshot of the activity at the end of a day, week, or month, then another picture at the end of the next period, and so on.
- Each periodic snapshots is stacked consecutively into the fact table.
- The periodic snapshot is the usual mechanism for easy retrieval of a regular, predictable, trendable view of the key business performance metrics.

Transaction and Periodic: same data!

- While transactions equate to little pieces of revenue, we can move easily from individual transactions to a daily snapshot by *summing the transactions*.
- In this situation, the periodic snapshot represents an *aggregation* of the transactional activity that occurred during that time period.
- Materialise the summary snapshot for performance reasons.
- Here, design of the snapshot table is closely related to the design of its companion transaction table.

Accumulating Snapshot Fact Tables

- An **accumulating snapshot** is a fact table that records a single row for an activity the enterprise tracks closely, such as a problem product or a single customer's credit.
- They represent an *indeterminate* time span, covering the complete life of a transaction or *discrete* product (or customer).
- Accumulating snapshots have multiple date stamps, representing the predictable major events or phases that take place during the course of a lifetime.
- Includes an additional date column to indicate *when* the snapshot row was last updated.

Updating the Data Warehouse

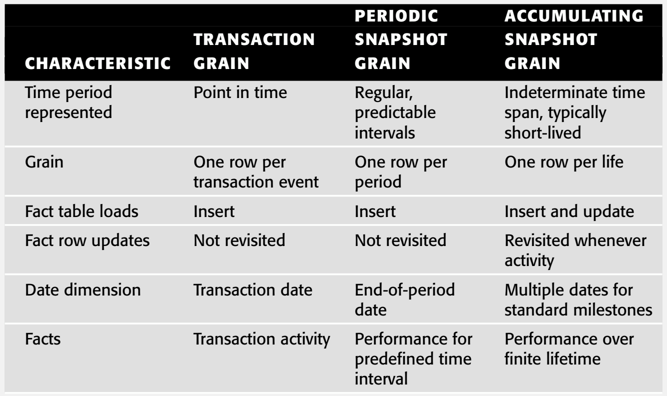
- Unlike other kinds of fact tables, the accumulating snapshot is intended to be *updated*.
- For example, dates are adjusted each time one of the milestones is reached.
- There may also be facts that track the number of days (or minutes) spent between each milestone.
- These *lags* in time are convenient: they can be computed from the dates.
- Building them into the fact table makes analysis much easier.

- Sometimes accumulating and periodic snapshots work in conjunction: build the monthly snapshot incrementally by adding the *effect* of each day's transactions to an accumulating snapshot.
- Consider the data warehouse as storing 36 months of historical data in the periodic snapshot, then the current rolling month would be month 37.
- Ideally, when the last day of the month has been reached, the accumulating snapshot simply becomes the new regular month in the time series, and a new accumulating snapshot is started the next day.

Transactions and Snapshots: the Holistic View

- Transactions and snapshots complement nicely in dimensional data warehouses.
- Used together, companion transaction and snapshot fact tables provide a complete view of the business.
- They *exist separately* as it is very difficult to combine these two contrasting perspectives.
- Data redundancy is not an issue: the goal is to publish data so that the organisation can analyse it effectively.

Summary



CHARACTERISTIC	TRANSACTION GRAIN	PERIODIC SNAPSHOT GRAIN	ACCUMULATING SNAPSHOT GRAIN
Time period represented	Point in time	Regular, predictable intervals	Indeterminate time span, typically short-lived
Grain	One row per transaction event	One row per period	One row per life
Fact table loads	Insert	Insert	Insert and update
Fact row updates	Not revisited	Not revisited	Revisited whenever activity
Date dimension	Transaction date	End-of-period date	Multiple dates for standard milestones
Facts	Transaction activity	Performance for predefined time interval	Performance over finite lifetime

Figure 17: Fact Table Comparison