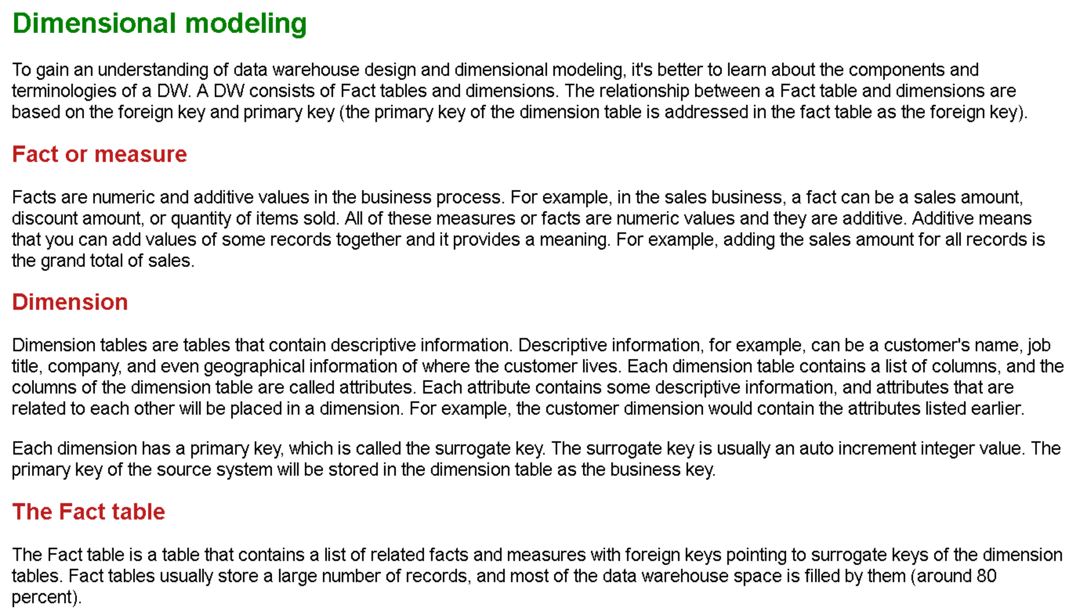
Topics

- Explain and discuss the concepts of OLAP, dimensions, axes, stars, and snowflakes.
- Design and build your first cube



Dimensional modeling

To gain an understanding of data warehouse design and dimensional modeling, it's better to learn about the components and terminologies of a DW. A DW consists of Fact tables and dimensions. The relationship between a Fact table and dimensions are based on the foreign key and primary key (the primary key of the dimension table is addressed in the fact table as the foreign key).

Fact or measure

Facts are numeric and additive values in the business process. For example, in the sales business, a fact can be a sales amount, discount amount, or quantity of items sold. All of these measures or facts are numeric values and they are additive. Additive means that you can add values of some records together and it provides a meaning. For example, adding the sales amount for all records is the grand total of sales.

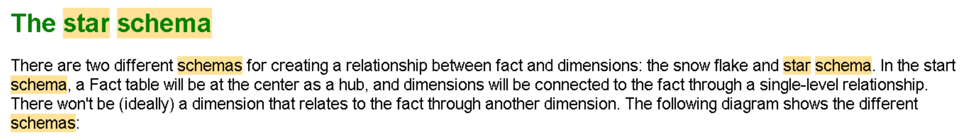
Dimension

Dimension tables are tables that contain descriptive information. Descriptive information, for example, can be a customer's name, job title, company, and even geographical information of where the customer lives. Each dimension table contains a list of columns, and the columns of the dimension table are called attributes. Each attribute contains some descriptive information, and attributes that are related to each other will be placed in a dimension. For example, the customer dimension would contain the attributes listed earlier.

Each dimension has a primary key, which is called the surrogate key. The surrogate key is usually an auto increment integer value. The primary key of the source system will be stored in the dimension table as the business key.

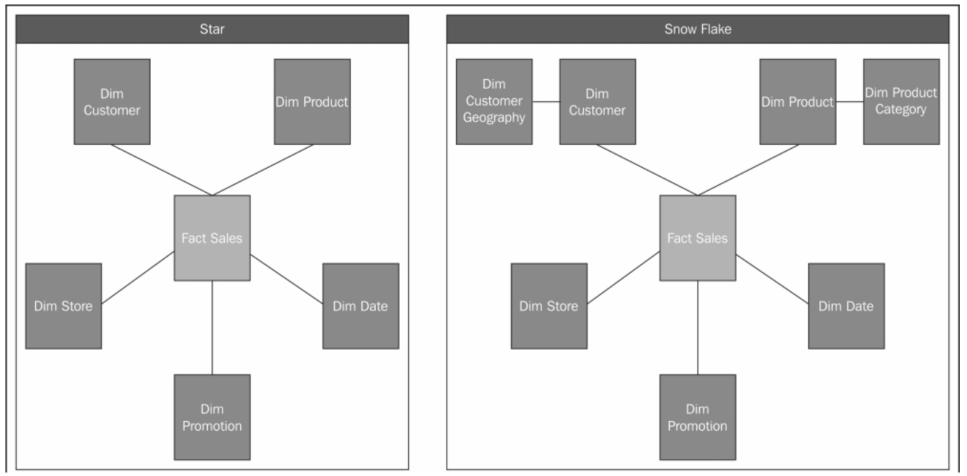
The Fact table

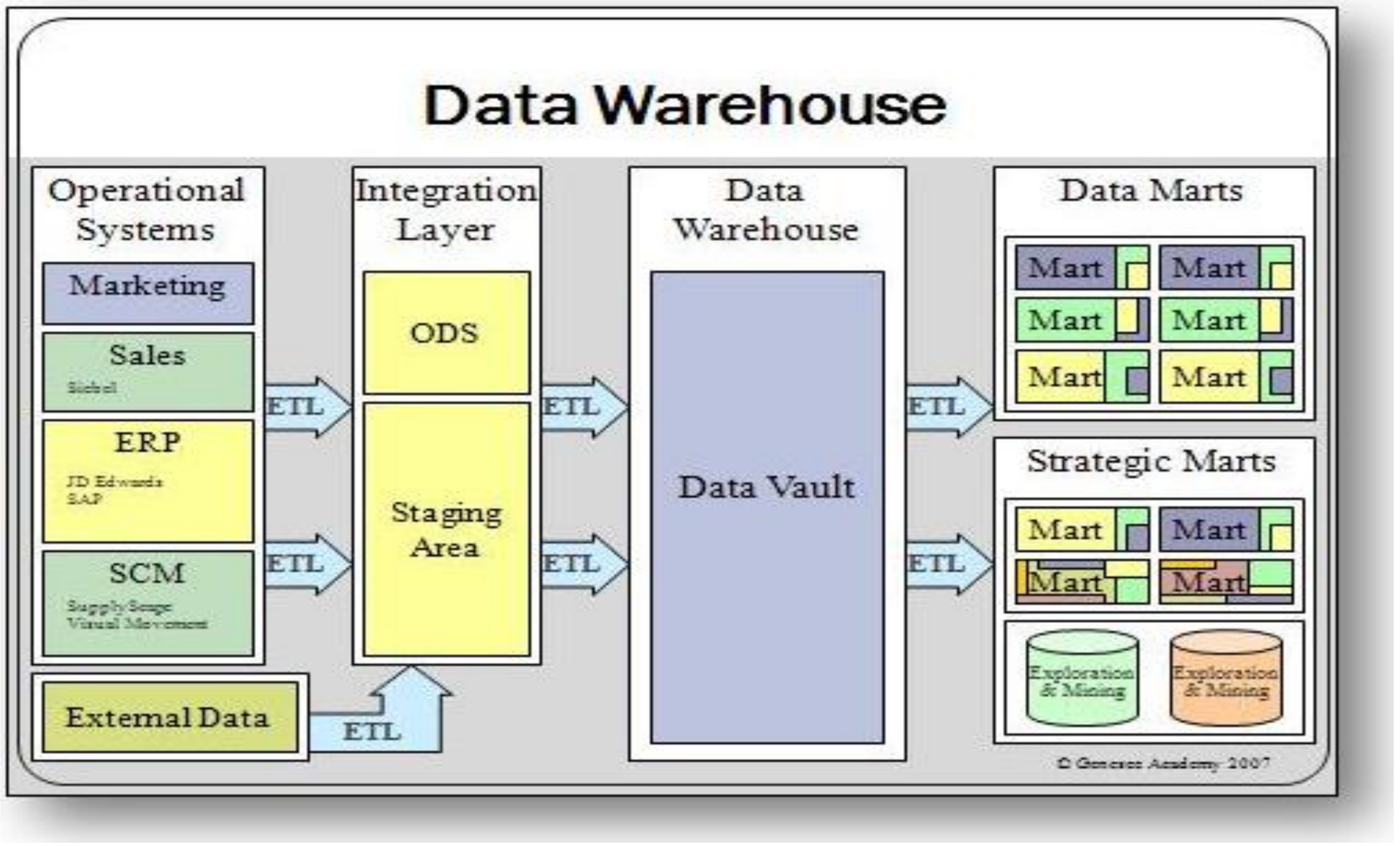
The Fact table is a table that contains a list of related facts and measures with foreign keys pointing to surrogate keys of the dimension tables. Fact tables usually store a large number of records, and most of the data warehouse space is filled by them (around 80 percent).



The star schema

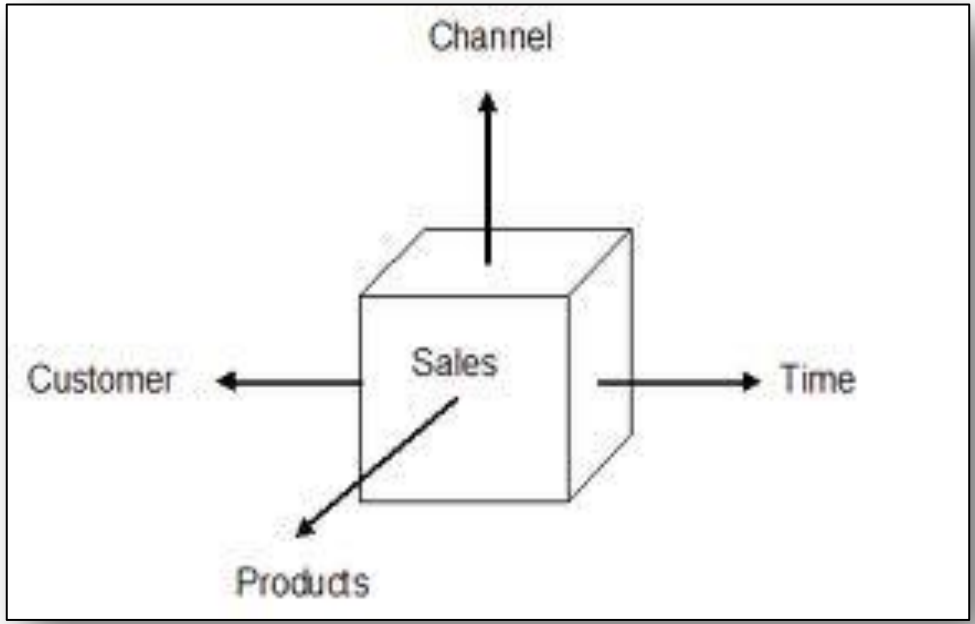
There are two different schemas for creating a relationship between fact and dimensions: the snow flake and star schema. In the start schema, a Fact table will be at the center as a hub, and dimensions will be connected to the fact through a single-level relationship. There won't be (ideally) a dimension that relates to the fact through another dimension. The following diagram shows the different schemas:




Data Warehouse
Operational Systems
Marketing
Sales
Siebel
ERP
JD Edwards
SAP
SCM
SupplyScape
Visual Movement
External Data
ETL
ETL
ETL
Integration Layer
ODS
Staging Area
ETL
ETL
Data Warehouse
Data Vault
ETL
ETL
Data Marts
Mart
Mart
Mart
Mart
Mart
Mart
Strategic Marts
Mart
Mart
Mart
Mart
Exploration & Mining
Exploration & Mining
© Genesee Academy 2007

Measures

- In a cube, a measure is a set of values that are based on a column in the cube's face table and are usually numeric.
- Measures are the central values of a cube that are analyzed.
- Measures are the numeric data of primary interest to end users browsing a cube. The measures you select depend on the types of information end users request. Some common measures are sales, cost, expenditures, or production count.



Rules of Fact Table Design

InternetSales from Assignment is an example of a fact table

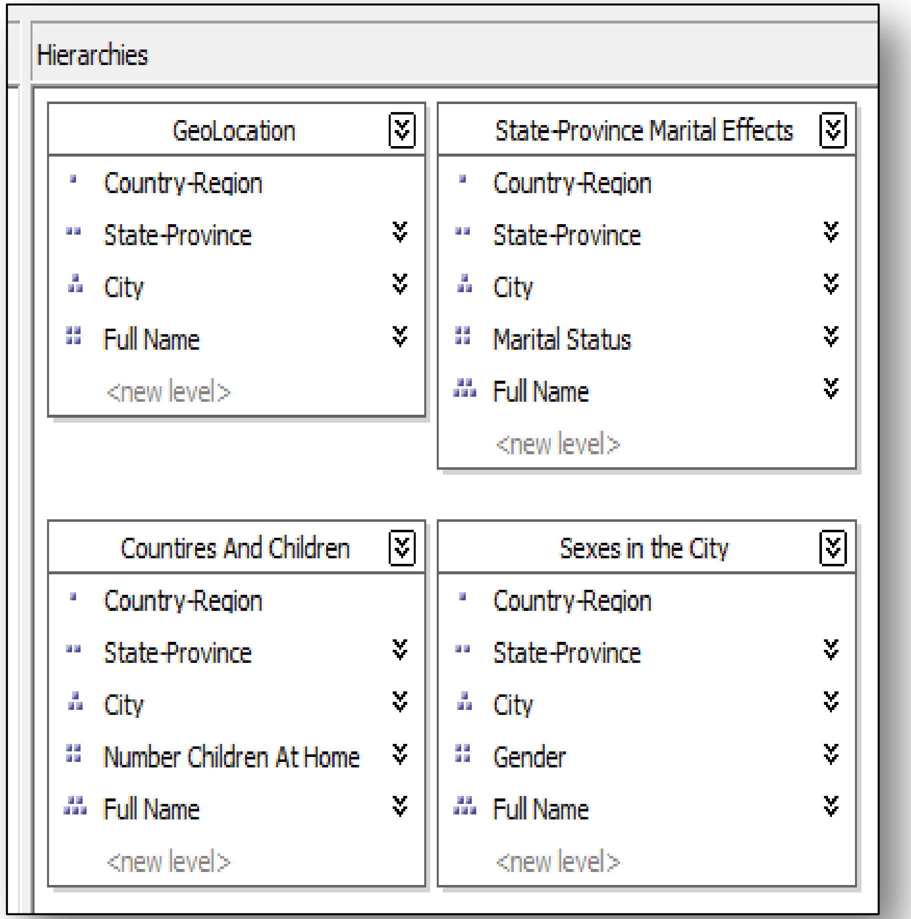
- The **Primary Key** of your fact table uses the minimum number columns possible & no surrogate keys.
  (It should be made up of FK’s and Degenerate Dimensions)
- **Referential Integrity** is a must. Every **foreign key** in the fact table must have a value.
- Avoid NULLs in the foreign key by using **flags** which are special values in place of null.
  - **Ex.** “No Shopper Card” in Customer Dimension
- The **granularity** of your fact table should be at the lowest, most detailed atomic grain captured by the business process.
- Each **fact** should be Additive, or re-designed to be as additive as possible.
- Each **fact** must be of the of the same granularity.
- Each fact table must reflect a single process that is modelled. Processes are not combined in a single fact table.

Hierarchies and Dimensions

- Hierarchies: A set of parent-child relationship, where a parent member summarizes its children.
- The end user can browse the cube by time period, geography, customer-base, product-line, information, or sales reason
- Every dimension excluding Internet Sales Order Details contains a set of hierarchies.

Example: Customer dimension contains four unique hierarchies

Example: The simplest hierarchy, "GeoLocation", can be applied to a row or column axis to help find this kind of data. The end-user can search in the "United States" for Country-Region, and then drill-down to State-Province and order the table data from "most sales" to "least sales"



Hierarchies

GeoLocation
- Country-Region
- State-Province
- City
- Full Name
- <new level>

State-Province Marital Effects
- Country-Region
- State-Province
- City
- Marital Status
- Full Name
- <new level>

Countires And Children
- Country-Region
- State-Province
- City
- Number Children At Home
- Full Name
- <new level>

Sexes in the City
- Country-Region
- State-Province
- City
- Gender
- Full Name
- <new level>

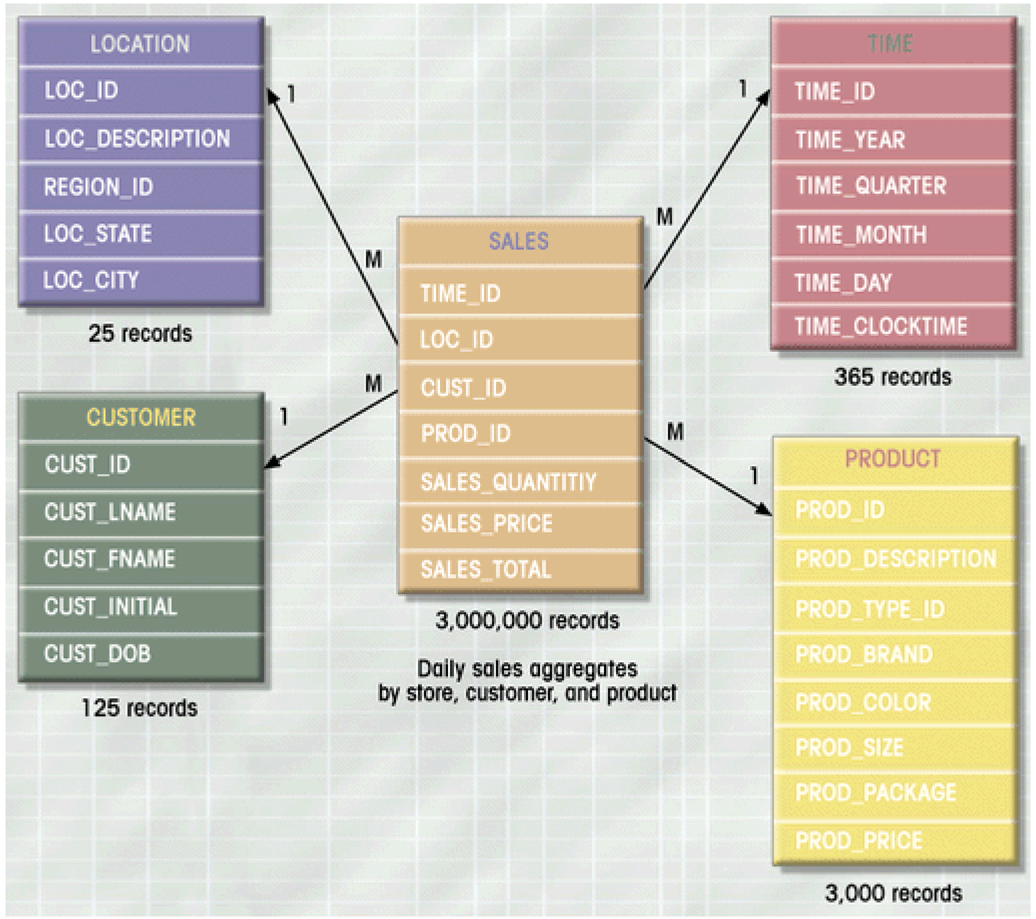
Dimensions

- A dimension is a broad grouping of related data about a major aspect of your business. For example, you have a dimension called, Products.
- A cube dimension can have only one cube hierarchy.
- In the example, we created five main dimensions in the cube: **Customer, Date, Promotion, Product, Sales Reason, and Internet Sales Order Details**.
- This combination of dimensions covers all aspects of Adventure Works

Types of Dimensions

- Role Dimension – Used on more than one axis e.g. Supply Date, Due Date, Order Date
- Conformed Dimension - allow facts and measures to be categorized and described in the same way across multiple facts and/or data marts ensuring consistent reporting across the enterprise.
- Junk Dimensions - number of miscellaneous, low-cardinality flags and indicators. Rather than making separate dimensions for each flag and attribute, you can create a single *junk dimension* combining them together.
- Degenerate Dimension - is a dimension key in the fact table that does not have its own dimension table

Structure of Dimension Table – Star Schema



Star Schema Representation

- Facts and dimensions are normally represented by physical tables in the data warehouse database.
- The fact table is related to each dimension table in a many-to-one (M:1) relationship.
- Fact and dimension tables are related by foreign keys and are subject to the primary/foreign key constraints.

Star Schema

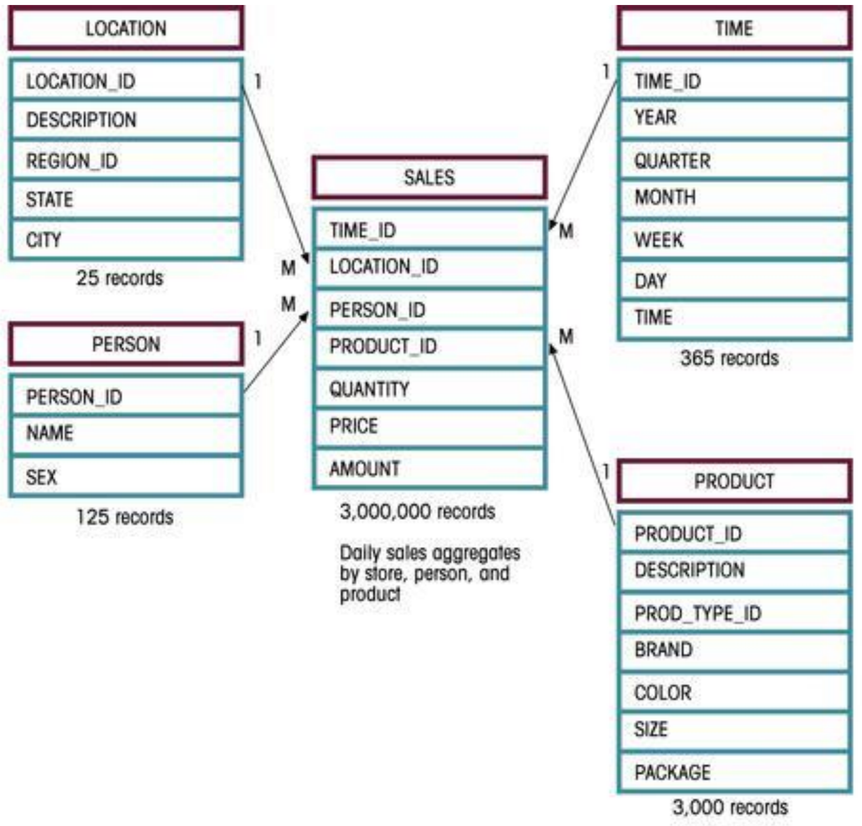


FIGURE 13.17 STAR SCHEMA FOR SALES

Star Schema separates Facts and Dimensions

- Facts, which hold the measurable, quantitative data about a business
- Dimensions which are descriptive attributes related to fact data.

Star Schema - Benefits

- Simpler queries - star schema join logic is generally simpler than the join logic required to retrieve data from a highly normalized transactional schema.
- Simplified business reporting logic - when compared to highly normalized schemas, the star schema simplifies common business reporting logic, such as period-over-period and as-of reporting.
- Query performance gains - star schemas can provide performance enhancements for read-only reporting applications when compared to highly normalized schemas.
- Fast aggregations - the simpler queries against a star schema can result in improved performance for aggregation operations.
- Feeding cubes - star schemas are used by all OLAP systems to build proprietary OLAP cubes efficiently; in fact, most major OLAP systems provide a ROLAP mode of operation which can use a star schema directly as a source without building a proprietary cube structure.

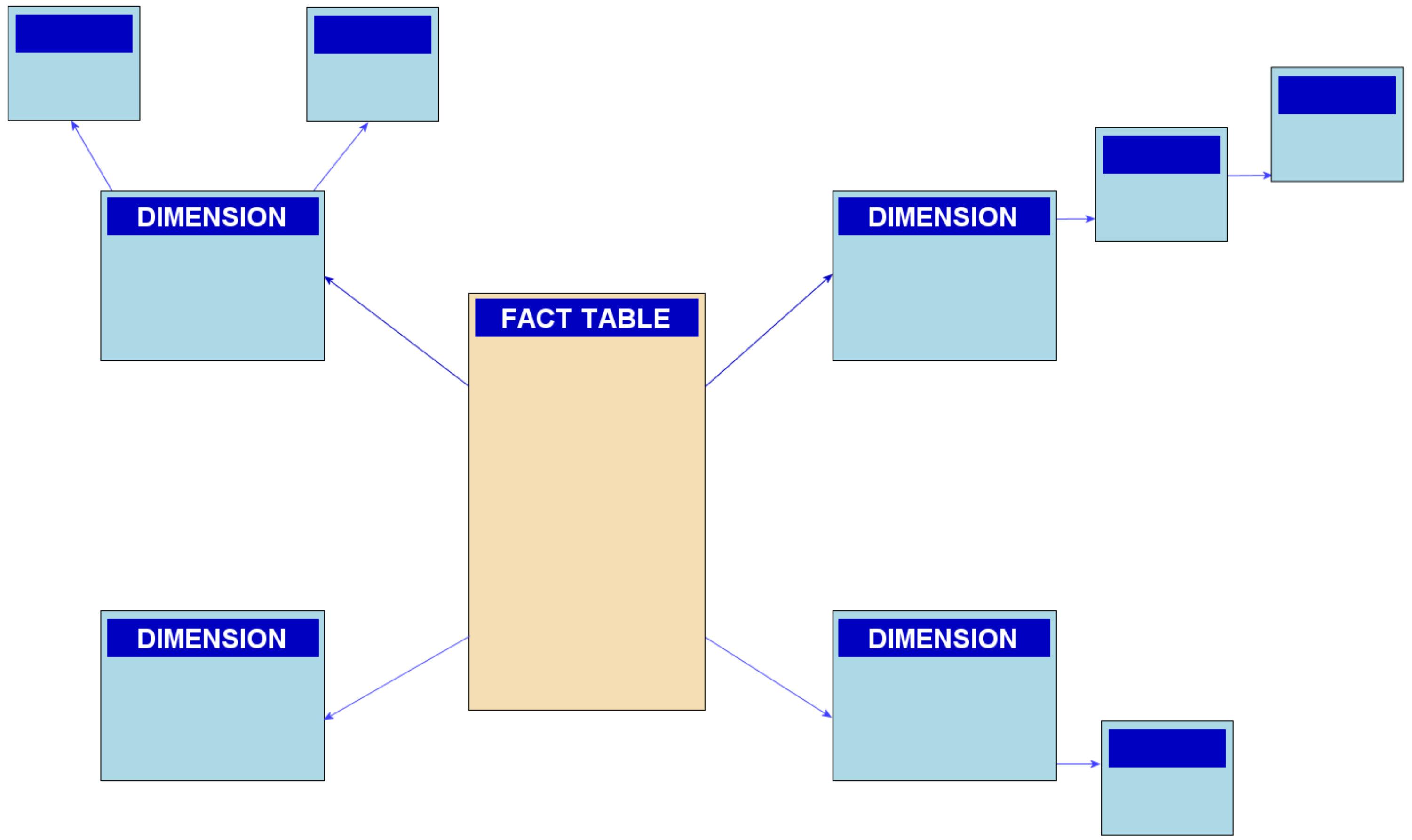
Star Schema Disadvantages

- Data Integrity is not enforced as well as it is in a highly normalized database. One-off inserts and updates can result in data anomalies which normalized schemas are designed to avoid.
- Loaded in a highly controlled fashion via batch processing or near-real time "trickle feeds", to compensate for the lack of protection afforded by normalization

Snowflake Schema

- Snowflaking is a method of normalizing the dimension tables in a star schema
- Similar to the star schema, but dimensions are normalized into multiple related tables.

Snowflake Schema



Benefits - Snowflake

- The snowflake schema is in the same family as the star schema logical model. In fact, the star schema is considered a special case of the snowflake schema. The snowflake schema provides some advantages over the star schema in certain situations, including:
- Some OLAP multidimensional database modeling tools are optimized for snowflake schemas.[2]
- Normalizing attributes results in storage savings, the tradeoff being additional complexity in source query joins.
- .

Disadvantages

- The primary disadvantage of the snowflake schema is that the additional levels of attribute normalization adds complexity to source query joins, when compared to the star schema.
- Snowflake schemas, in contrast to flat single table dimensions, have been heavily criticised. Their goal is assumed to be an efficient and compact storage of normalised data but this is at the significant cost of poor performance when browsing the joins required in this dimension.[3] This disadvantage may have reduced in the years since it was first recognized, owing to better query performance within the browsing tools.
- When compared to a highly normalized transactional schema, the snowflake schema's denormalization removes the data integrity assurances provided by normalized schemas.[*citation needed*] Data loads into the snowflake schema must be highly controlled and managed to avoid update and insert anomalies

Rules of Dimension Table Design

- Verbose attribute values should be as descriptive as possible.
- Descriptive columns – should be easy to tell what the column means.
- Complete – no null / empty values in any of the attributes.
- Discretely valued – one business entity value per row.
- Quality Assured – data is clean and consistent.
- Should always contain a business key, or legacy PK from source system.
- Always have a **Surrogate Primary Key.** You do not introduce a dependency on an external key.

What's Wrong w/This Dimension of Products?

Prod Id	Prod Name	Prod Cat	Prod Price	Prod Region Code
A	Apple	Fruit	$2.00	E
B	Carrot	Veg	$1.50	S
C	Cherries	Friut	$3.00	S
D	Lettuce	Veg	$1.50	
E	Apple	Fruit	$2.00	E

Can you find the 6 things wrong with the implementation of this dimension?

1. No surrogate key
2. Not discretely values
3. Poor data quality
4. Incomplete / missing data
5. Poor descriptions
6. Non-verbose data

What's Wrong w/This Dimension?

No Surrogate Key

Poor Descriptions

Prod Id	Prod Name	Prod Cat	Prod Price	Prod Reg Code
A	Apple	Fruit	$2.00	E
B	Carrot	Veg	$1.50	S
C	Cherries	Friut	$3.00	S
D	Lettuce	Veg	$1.50	
E	Apple	Fruit	$2.00	E

Not Verbose (What do S & E mean?)

Not Discretely Valued

Poor Data Quality

Incomplete

Types of Measures

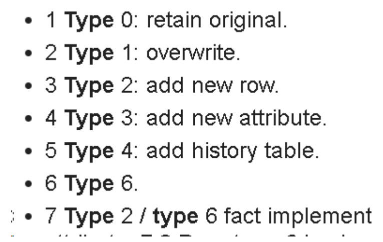
- Additive Can be aggregated over all dimensions
  Example: sales price
  Often occur in event facts
- Semi-additive

Cannot be aggregated over some dimensions - typically time
  Example: inventory
  Often occur in snapshot facts

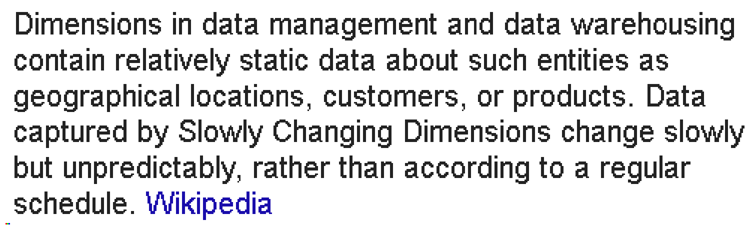
- Non-additive
  Cannot be aggregated over any dimensions
  Example: average sales price
  Occur in all types of facts

Slowly Changing Dimension Techniques

there are 12 ways, we will stick with 0, 1, 2, and 3



- 1 **Type** 0: retain original.
- 2 **Type** 1: overwrite.
- 3 **Type** 2: add new row.
- 4 **Type** 3: add new attribute.
- 5 **Type** 4: add history table.
- 6 **Type** 6.
- 7 **Type** 2 / **type** 6 fact implement



Dimensions in data management and data warehousing contain relatively static data about such entities as geographical locations, customers, or products. Data captured by Slowly Changing Dimensions change slowly but unpredictably, rather than according to a regular schedule. Wikipedia

- Type 0: Retain Original With slowly changing dimension type 0, the dimension attribute value never changes, so facts are always grouped by this original value. Type 0 is appropriate for any attribute labeled “original,” such as a customer’s original credit score or a durable identifier. It also applies to most attributes in a date dimension.
-
- Type 1: Overwrite With slowly changing dimension type 1, the old attribute value in the dimension row is overwritten with the new value; type 1 attributes always reflects the most recent assignment, and therefore this technique destroys history. Although this approach is easy to implement and does not create additional dimension rows, you must be careful that aggregate fact tables and OLAP cubes affected by this change are recomputed.
-
- Type 2: Add New Row Slowly changing dimension type 2 changes add a new row in the dimension with the updated attribute values. This requires generalizing the primary key of the dimension beyond the natural or durable key because there will potentially be multiple rows describing each member. When a new row is created for a dimension member, a new primary surrogate key is assigned and used as a foreign key in all fact tables from the moment of the update until a subsequent change creates a new dimension key and updated dimension row. A minimum of three additional columns should be added to the dimension row with type 2 changes: 1) row effective date or date/time stamp; 2) row expiration date or date/time stamp; and 3) current row indicator.
-
- Type 3: Add New Attribute Slowly changing dimension type 3 changes add a new attribute in the dimension to preserve the old attribute value; the new value overwrites the main attribute as in a type 1 change. This kind of type 3 change is sometimes called an alternate reality. A business user can group and filter fact data by either the current value or alternate reality. This slowly changing dimension technique is used relatively infrequently.

MDX

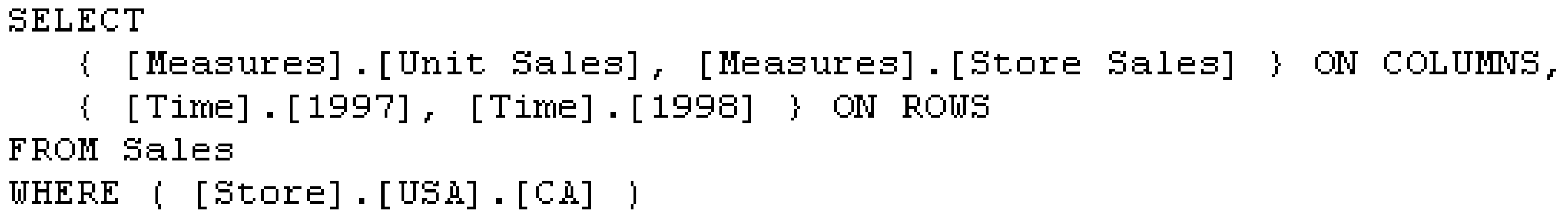
(Query language from hell…)

MDX – is the query language for the DW structures

- The DW structure is regarded as a Cube where the dimensions are viewed as the axes

MDX (Multidimensional Expressions)

- Query language for a cube
- Similar but different from SQL
- Handles DML as well as DDL
- Basic format is:



```
SELECT
   { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMNS,
   { [Time].[1997], [Time].[1998] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

MDX

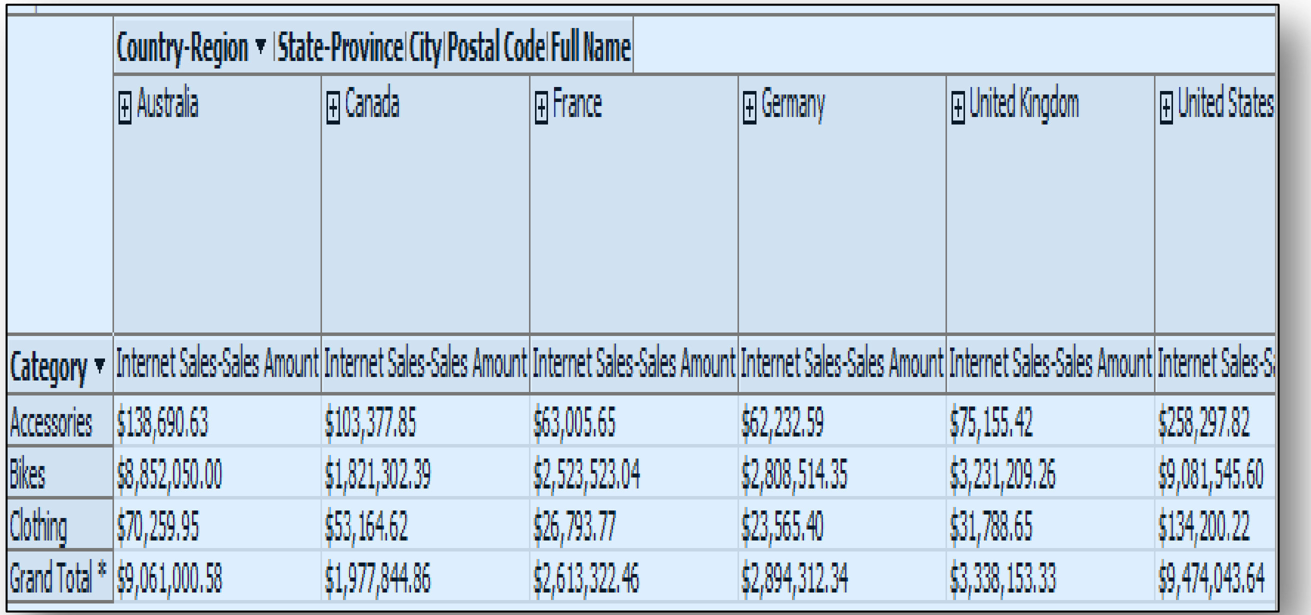
- Members, tuples, and sets (Oh My!)
- Axis dimensions
  - Columns, rows, pages, sections, chapters
  - Axis(n)
- Slicer dimensions
  - Where (<tuple definition>)
- MDX functions
  - Let’s not go there tonight…

Slice/Dice/Roll-up/Drill-Down/Pivot

SLICE:

•Is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension.

•It is made up of five layers, the top is the most general filter, grouped by its Country/Region



	Country-Region ▾ | State-Province | City | Postal Code | Full Name					
	⊞ Australia	⊞ Canada	⊞ France	⊞ Germany	⊞ United Kingdom	⊞ United States
Category ▾	Internet Sales-Sales Amount	Internet Sales-Sales Amount	Internet Sales-Sales Amount	Internet Sales-Sales Amount	Internet Sales-Sales Amount	Internet Sales-S
Accessories	$138,690.63	$103,377.85	$63,005.65	$62,232.59	$75,155.42	$258,297.82
Bikes	$8,852,050.00	$1,821,302.39	$2,523,523.04	$2,808,514.35	$3,231,209.26	$9,081,545.60
Clothing	$70,259.95	$53,164.62	$26,793.77	$23,565.40	$31,788.65	$134,200.22
Grand Total *	$9,061,000.58	$1,977,844.86	$2,613,322.46	$2,894,312.34	$3,338,153.33	$9,474,043.64

Example of a “slice” of the cube

Roll-Up/Drill-Down

Roll-up

- A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as:

  "profit = sales - expenses"

Drill-Down/ Drill-Up

- Is a specific analytical technique where the user navigates among levels of data ranging from the most summarized (up) to the most detailed (down).
- The drilling paths may be defined by the hierarchies within dimensions or other relationships that may be dynamic within or between dimensions.

Example: The "Sexes in the City" hierarchy can be used when browsing the cube

The country would be set to "Canada", then drill-down to the province of Ontario, then drill-down again to the city of "Toronto", and then finally drill-down to just "Females" in Toronto

Dice/ Pivot

Dice

- The dice operation produces a sub cube by allowing the analyst to pick specific values of multiple dimensions.

Pivot

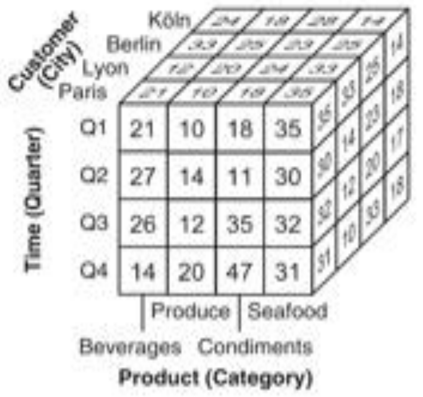
- Pivot allows an analyst to rotate the cube in space to see its various faces.

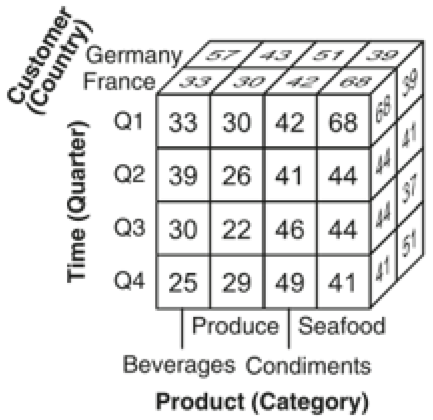
Example:

- “Cities” could be arranged vertically and products horizontally while viewing data for a particular quarter. All these techniques are often combined to help squeeze as much information as possible from the cube, both quickly and efficiently.

Roll up the original cube to the country level

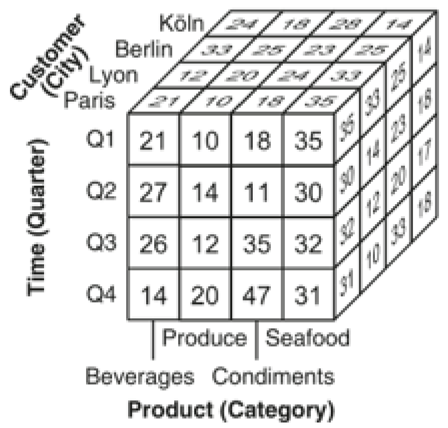
a



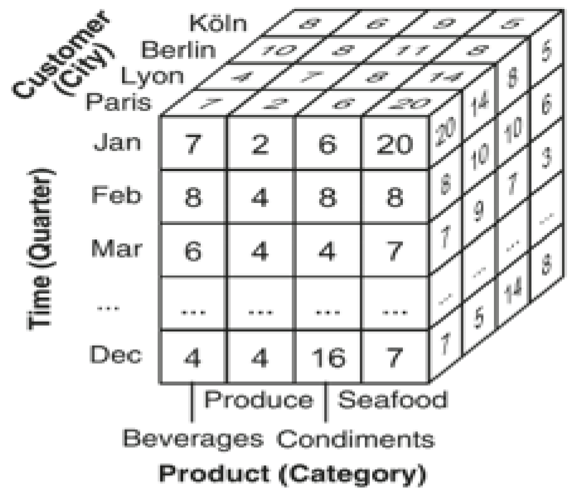


Drill Down to the month level

a

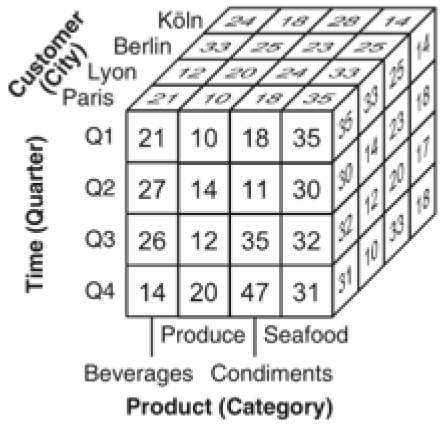


c

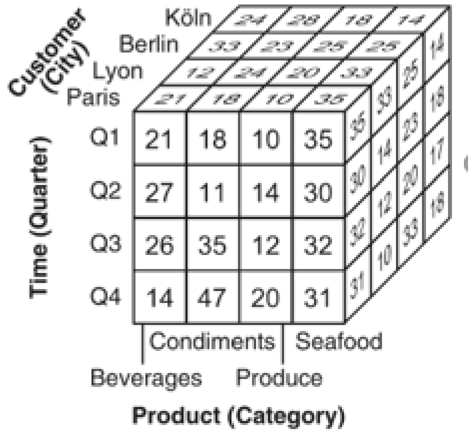


Sort by Product Name

a

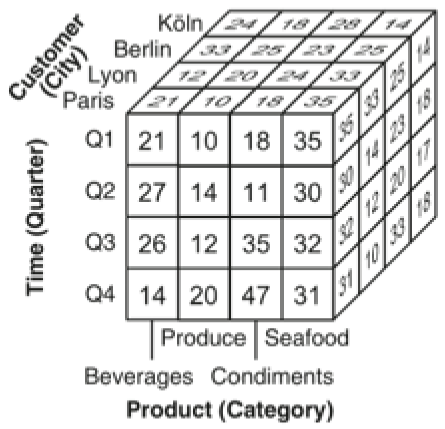


d

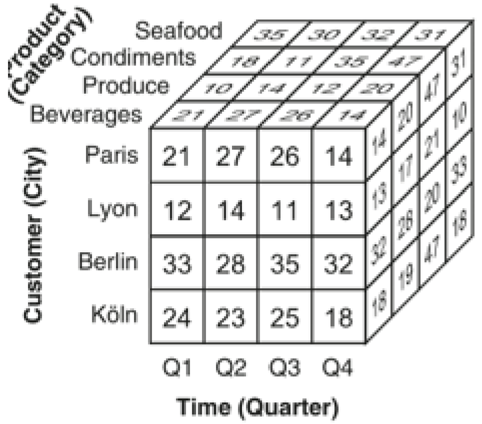


Pivot the cube

a

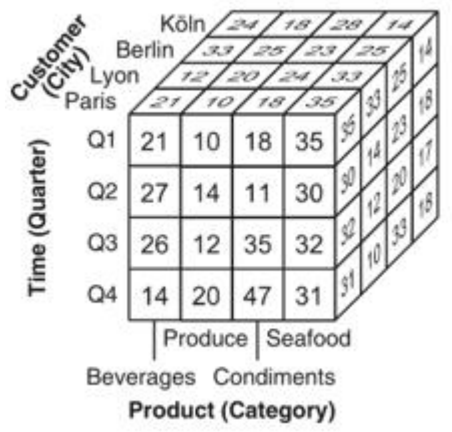


e

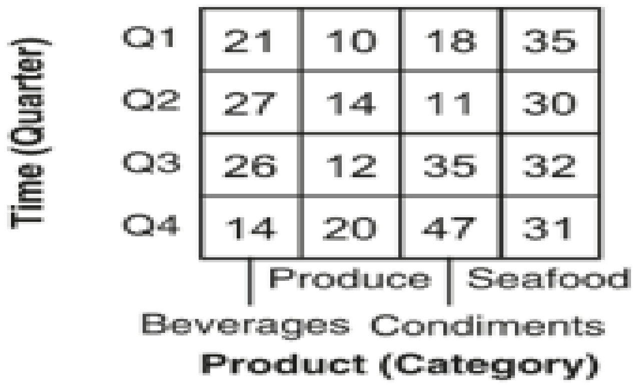


Slice on City = Paris

a



f

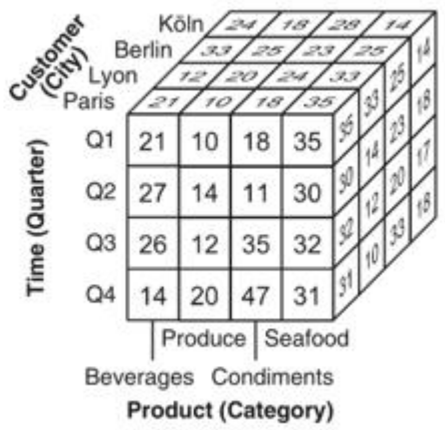


Time (Quarter)	Beverages	Produce	Condiments	Seafood
Q1	21	10	18	35
Q2	27	14	11	30
Q3	26	12	35	32
Q4	14	20	47	31

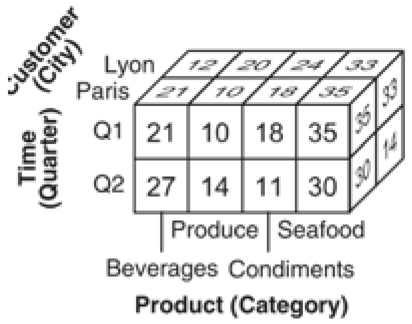
Product (Category)

Slice on City = Paris or Lyon and Quarter = Q1 or Q2

a

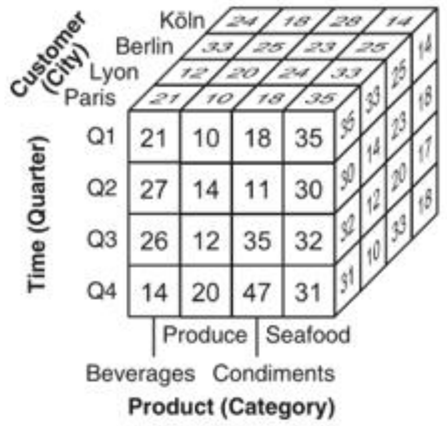


g

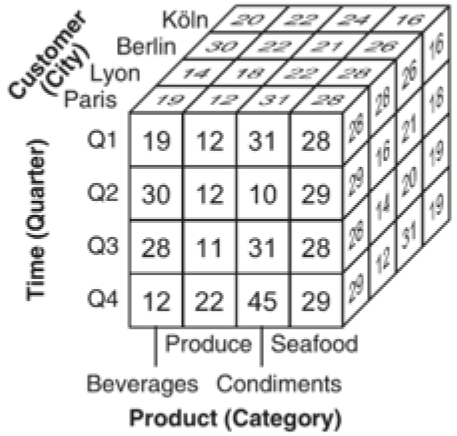


Cube for 2011

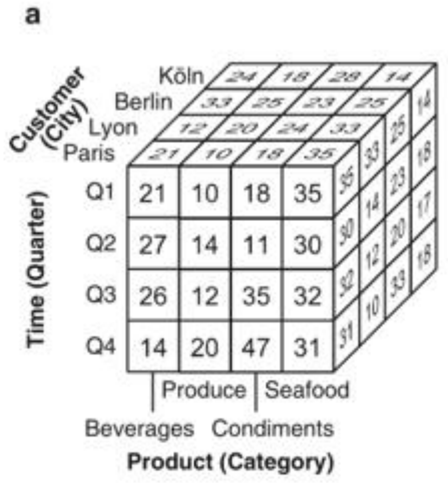
a

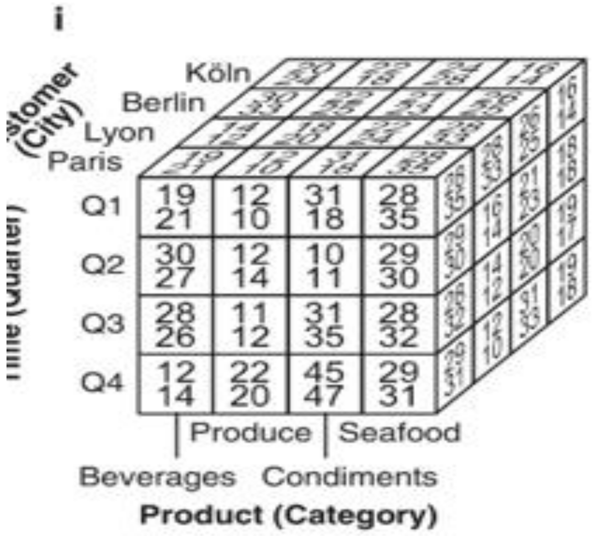


h



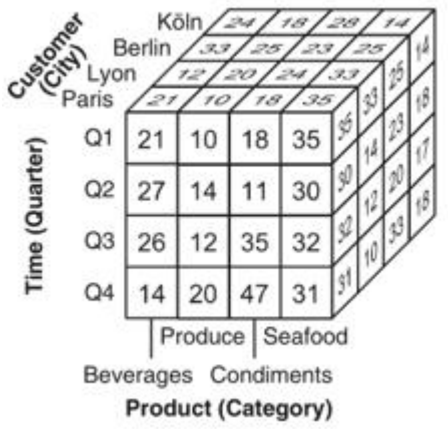
Drill across operation



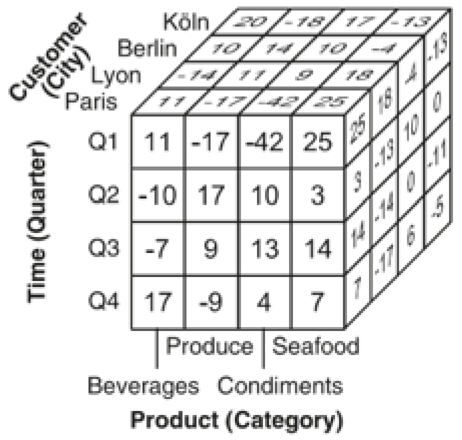


Percentage change over the year

a

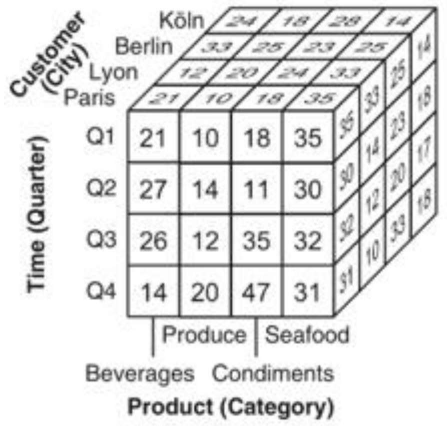


j



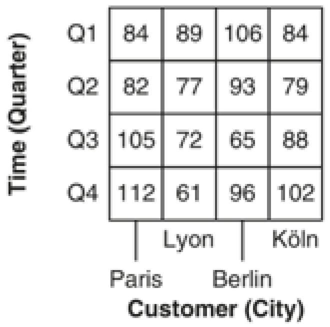
Total sales by quarter and city

a



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

k



Time (Quarter)	Paris	Lyon	Berlin	Köln
Q1	84	89	106	84
Q2	82	77	93	79
Q3	105	72	65	88
Q4	112	61	96	102

Customer (City)

MDX Examples -

```
SELECT { [Measures].[Sales Amount], [Measures].[Tax Amount] } ON COLUMNS, {
[Date].[Fiscal].[Fiscal Year].&[2002], [Date].[Fiscal].[Fiscal Year].&[2003] } ON ROWS FROM
[Adventure Works] WHERE ( [Sales Territory].[Southwest] )
```

- SELECT { [Measures].[Sales Amount], [Measures].[Tax Amount] } ON 0, { [Date].[Fiscal].[Fiscal Year].&[2002], [Date].[Fiscal].[Fiscal Year].&[2003] } ON 1 FROM [Adventure Works] WHERE ([Sales Territory].[Southwest])

MDX - Slicer

- SELECT {[Measures].[Internet Sales Amount]} ON COLUMNS, NON EMPTY {[Date].[Calendar].MEMBERS} ON ROWS FROM [Adventure Works]

Slicer Restriction with where

```
 SELECT {[Measures].[Internet Sales Amount]} ON COLUMNS,
[Date].[Calendar Year].MEMBERS ON ROWS
FROM [Adventure Works]
```

- Adding a WHERE clause, as follows:

```
 SELECT {[Measures].[Internet Sales Amount]} ON COLUMNS,
[Date].[Calendar Year].MEMBERS ON ROWS
FROM [Adventure Works]
WHERE([Customer].[Customer Geography].[Country].&[United States])
```